(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **10320205 A**

(43) Date of publication of application: **04.12.98**

(51) Int. Cl **G06F 9/44**
G06F 9/44

(21) Application number: **09130160**

(22) Date of filing: **20.05.97**

(71) Applicant: **FUJITSU LTD**

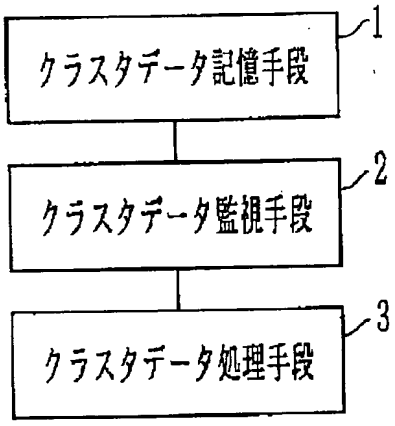
(72) Inventor: **NISHIGAYA TAKESHI**
IIDA ICHIRO

(54) INFORMATION PROCESSOR

(57) Abstract:

PROBLEM TO BE SOLVED: To delete or substitute a part of a program code without stopping a program in operating.

SOLUTION: A class data monitoring means 2 monitors the using state of class data stored in a class data storing means 1 in each added class data and a class data processing means 3 deletes or substitutes a part of a program during the operation of the program by deleting or substituting class data based on the using state of the class data monitored by the means 2.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－320205

(43)公開日　平成10年(1998)12月４日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/44	５３０	Ｇ０６Ｆ　9/44	５３０Ｍ
	５５２		５５２

審査請求　未請求　請求項の数44　ＯＬ　（全 43 頁）

(21)出願番号　特願平9－130160

(22)出願日　平成９年(1997)５月20日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　西ヶ谷　岳
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(72)発明者　飯田　一朗
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(74)代理人　弁理士　大菅　義之　（外１名）

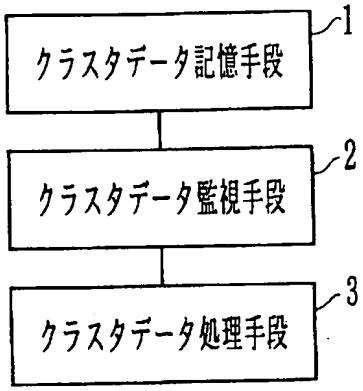
(54)【発明の名称】　情報処理装置

(57)【要約】

【課題】　動作中のプログラムを停止させることなく、そのプログラムコードの一部を削除したり、置き換えたりすることを可能とする。

【解決手段】　クラスデータ監視手段２は、クラスデータ記憶手段１に記憶されているクラスデータの使用状況を追加されたクラスデータごとに監視し、クラスデータ処理手段３は、クラスデータ監視手段２で監視されているクラスデータの使用状況に基づいて、クラスデータの削除や置換を行うことにより、プログラムの動作中に、そのプログラムの一部の削除や置換を行えるようにする。

本発明の一実施例に係わる
情報処理装置の機能的な構成を示すブロック図



1

【特許請求の範囲】

【請求項１】　オブジェクトに追加されたクラスデータを記憶するクラスデータ記憶手段と、
前記クラスデータの使用状況を監視するクラスデータ監視手段と、
前記クラスデータの使用状況に基づいて、前記クラスデータの処理を行うクラスデータ処理手段とを備えることを特徴とする情報処理装置。

【請求項２】　前記クラスデータ処理手段は、
前記クラスデータについての処理要求を受け付ける受付手段と、
前記処理要求のあったクラスデータが使用中かどうかを判定する判定手段と、
前記クラスデータが使用中の場合、前記クラスデータの使用が終了するまで、前記クラスデータの処理を延期する延期手段とを備えることを特徴とする請求項１に記載の情報処理装置。

【請求項３】　前記クラスデータ処理手段は、
前記クラスデータについての処理要求を受け付ける受付手段と、
前記処理要求のあったクラスデータを使用しているオブジェクトに中止要求を発行する中止要求発行手段と、
前記オブジェクトが前記中止要求に応答した後、前記クラスデータの処理を行う実行手段とを備えることを特徴とする請求項１または２に記載の情報処理装置。

【請求項４】　前記受付手段は、前記クラスデータについての処理要求を受け付けるかどうかを検査する検査手段をさらに備えることを特徴とする請求項２または３に記載の情報処理装置。

【請求項５】　前記クラスデータ処理手段は、前記クラスデータの追加要求単位で前記クラスデータを前記クラスデータ記憶手段に追加するクラスデータ追加手段を備えることを特徴とする請求項１～４のいずれか１項に記載の情報処理装置。

【請求項６】　前記クラスデータ処理手段は、削除対象のクラスデータの使用が終了してから、前記クラスデータを削除するクラスデータ削除手段を備えることを特徴とする請求項１～５のいずれか１項に記載の情報処理装置。

【請求項７】　前記クラスデータ処理手段は、置換対象のクラスデータの使用が終了してから、前記クラスデータを置換するクラスデータ置換手段を備えることを特徴とする請求項１～６のいずれか１項に記載の情報処理装置。

【請求項８】　クラスデータの使用状況を監視するエージェントクラスオブジェクトを生成するためのエージェントクラスを格納するエージェントクラス格納手段と、
前記クラスデータに対する処理が行われる場合、前記エージェントクラスに基づいて前記エージェントクラスオブジェクトを生成するエージェントクラスオブジェクト生成手段とを備えることを特徴とする情報処理装置。

2

【請求項９】　前記エージェントクラスは、
前記エージェントクラスオブジェクトに追加されたクラスデータの管理を行うクラス管理オブジェクトと、
前記クラス管理オブジェクトで管理されるクラスデータを記憶する共有クラスデータテーブルと、
前記クラスデータに対する処理を実行する外部公開メソッドとを備えることを特徴とする請求項８に記載の情報処理装置。

【請求項１０】　前記エージェントクラスは、
前記クラスデータの追加要求に対して生成されたクラス管理オブジェクトと前記クラスデータの追加要求時に与えられた名前との対応関係を記憶したクラス管理オブジェクトテーブルをさらに備えることを特徴とする請求項９に記載の情報処理装置。

【請求項１１】　前記クラス管理オブジェクトは、
各クラス管理オブジェクトに固有のクラスデータを格納する局所クラスデータテーブルと、
前記クラスデータの削除要求を記憶する削除フラグと、
前記クラスデータの置換要求時の新たなクラスデータを記憶する置換クラスデータ記憶領域と、
前記クラスデータを実行しているスレッドの識別情報を記憶する実行中スレッドリストと、
前記クラスデータに対応するクラス名を記憶するクラス名リストと、
前記クラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する参照オブジェクトリストと、
前記クラスデータの参照元のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する被参照オブジェクトリストとをさらに備えることを特徴とする請求項１０に記載の情報処理装置。

【請求項１２】　前記エージェントクラスは、他のエージェントクラスオブジェクトからのアクセス権をチェックするアクセス制御クラスを前記共有クラスデータテーブルに備えることを特徴とする請求項９～１１のいずれか１項に記載の情報処理装置。

【請求項１３】　前記エージェントクラスオブジェクトは、
他のエージェントクラスオブジェクトからのアクセスがあった場合、前記アクセス制御クラスに基づいてアクセス制御オブジェクトを生成するアクセス制御オブジェクト生成手段と、
前記アクセス制御オブジェクトに基づいて、前記アクセス権のチェックを実行するアクセス制御オブジェクト実行手段とを備えることを特徴とする請求項１２に記載の情報処理装置。

【請求項１４】　前記外部公開メソッドは、前記エージェントクラスオブジェクトにクラスデータを追加するクラスデータ追加手段を備えることを特徴とする請求項１

１に記載の情報処理装置。

【請求項１５】　前記クラスデータ追加手段は、

前記エージェントクラスオブジェクトに追加されたクラスデータを管理するクラス管理オブジェクトを生成するクラス管理オブジェクト生成手段と、

前記クラス管理オブジェクト内のクラス名リストに前記クラスデータのクラス名を登録するクラス名登録手段と、

前記クラスデータを前記共有クラスデータテーブルに書き込むクラスデータ書き込み手段と、

前記クラスデータの追加要求時に与えられた名前を前記クラス管理オブジェクトテーブルに登録するクラス管理オブジェクト登録手段とを備えることを特徴とする請求項１４に記載の情報処理装置。

【請求項１６】　前記クラスデータ追加手段は、

前記エージェントクラスオブジェクトに追加されたクラスデータのクラス名が、前記共有クラスデータテーブルに登録されているクラスデータのクラス名と重複する場合、前記エージェントクラスオブジェクトに追加されたクラスデータを前記共有クラスデータテーブルから削除することにより、前記共有クラスデータテーブルの元の状態を復元する共有クラスデータテーブル復元手段を備えることを特徴とする請求項１５に記載の情報処理装置。

【請求項１７】　前記外部公開メソッドは、前記エージェントクラスオブジェクトに保有されているクラスデータを実行するクラスデータ実行手段を備えることを特徴とする請求項１３～１６のいずれか１項に記載の情報処理装置。

【請求項１８】　前記クラスデータ実行手段は、

実行中のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストに、前記クラスデータを実行しているスレッドの識別情報を設定する第１スレッドリスト設定手段と、

実行中のクラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストに、前記クラスデータを実行しているスレッドの識別情報を設定する第２スレッドリスト設定手段とを備えることを特徴とする請求項１７に記載の情報処理装置。

【請求項１９】　前記クラスデータ実行手段は、

クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されているかどうかを検索する第１実行中スレッドリスト検索手段と、

前記クラス管理オブジェクトの削除フラグの設定状態を判定する削除フラグ判定手段と、

前記実行中スレッドリストが空で、かつ、前記削除フラグがオンのクラス管理オブジェクトが存在している場合、前記クラス管理オブジェクトのクラス名リストのクラス名に対応するクラスデータを前記共有クラスデータテーブルから削除する削除実行手段とを備えることを特

徴とする請求項１８に記載の情報処理装置。

【請求項２０】　前記クラスデータ実行手段は、

クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されているかどうかを検索する第２実行中スレッドリスト検索手段と、

前記クラス管理オブジェクトの置換クラスデータ記憶領域を検索する置換クラスデータ検索手段と、

前記実行中スレッドリストが空で、かつ、前記置換クラスデータ記憶領域にクラスデータが記憶されているクラス管理オブジェクトが存在している場合、前記共有クラスデータテーブルに記憶されている置換対象のクラスデータを、前記置換クラスデータ記憶領域に記憶されているクラスデータで置換する置換実行手段とを備えることを特徴とする請求項１８または１９に記載の情報処理装置。

【請求項２１】　前記外部公開メソッドは、前記エージェントクラスオブジェクトからクラスデータを削除するクラスデータ削除手段を備えることを特徴とする請求項１３～２０のいずれか１項に記載の情報処理装置。

【請求項２２】　前記クラスデータ削除手段は、

削除対象のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストを検索する第３実行中スレッドリスト検索手段と、

前記実行中スレッドリストにスレッドの識別情報が設定されている場合、前記クラスデータの削除を中止し、前記クラス管理オブジェクトの削除フラグをオンに設定する削除フラグ設定手段とを備えることを特徴とする請求項２１に記載の情報処理装置。

【請求項２３】　前記外部公開メソッドは、前記エージェントクラスオブジェクトのクラスデータを置換するクラスデータ置換手段を備えることを特徴とする請求項１３～２２のいずれか１項に記載の情報処理装置。

【請求項２４】　前記クラスデータ置換手段は、

置換対象のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストを検索する第４実行中スレッドリスト検索手段と、

前記実行中スレッドリストにスレッドの識別情報が設定されている場合、前記クラスデータの置換を中止し、前記クラス管理オブジェクトの置換クラスデータ記憶領域に新たなクラスデータを設定する置換データ設定手段とを備えることを特徴とする請求項２３に記載の情報処理装置。

【請求項２５】　前記クラス管理オブジェクトの被参照オブジェクトリストが空でない場合、エラーを通知するエラー通知手段と、

前記クラス管理オブジェクトの参照オブジェクトリストが空でない場合、削除対象のクラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの被参照オブジェクトリストから、前記クラス管理オブジェクトの識別情報を削除する識別情報削除手段とを備える

ことを特徴とする請求項２１～２４のいずれか１項に記載の情報処理装置。

【請求項２６】　前記外部公開メソッドは、他のエージェントクラスオブジェクトから送られたクラスデータを実行するクラスデータ遠隔実行手段を備えることを特徴とする請求項１３～２５のいずれか１項に記載の情報処理装置。

【請求項２７】　前記クラスデータ遠隔実行手段は、クラス管理オブジェクトの局所クラスデータテーブルに前記クラスデータを設定するクラスデータ設定手段を備えることを特徴とする請求項２６に記載の情報処理装置。

【請求項２８】　前記クラスデータ遠隔実行手段は、
クラスデータ及び前記クラスデータから生成されたオブジェクトのシリアライズデータを受信するデータ受信手段と、
前記クラスデータ及び前記シリアライズデータに基づいて、前記クラスデータから生成されたオブジェクトを復元するオブジェクト復元手段とを備えることを特徴とする請求項２６または２７に記載の情報処理装置。

【請求項２９】　前記クラスデータ遠隔実行手段は、
実行対象のクラスデータが参照しているクラスデータを前記共有クラスデータテーブルから呼び出すクラスデータ呼び出し手段と、
前記実行対象のクラスデータとともに前記実行対象のクラスデータが参照しているクラスデータを、他のエージェントクラスオブジェクトに送信する第１クラスデータ送信手段とを備えることを特徴とする請求項２６～２８のいずれか１項に記載の情報処理装置。

【請求項３０】　前記クラスデータ遠隔実行手段は、
実行対象のクラスデータが、送信先のエージェントクラスオブジェクトの共有クラスデータテーブルに存在しているかどうかを検索する重複クラス検索手段と、
前記検索結果に基づいて、送信先のエージェントクラスオブジェクトに存在しているクラスデータを除外してから、前記送信先のエージェントクラスオブジェクトに実行対象のクラスデータを送信する第２クラスデータ送信手段とを備えることを特徴とする請求項２６～２９のいずれか１項に記載の情報処理装置。

【請求項３１】　プログラムを記憶するプログラム記憶手段と、
前記プログラムの使用状況を前記プログラムの機能ごとに監視するプログラム監視手段と、
前記プログラムの機能の使用状況に基づいて、前記プログラムの機能に対する処理を行うプログラム処理手段とを備えることを特徴とする情報処理装置。

【請求項３２】　前記プログラム監視手段は、前記プログラムの機能を特定するための名前を管理する名前管理手段を備えることを特徴とする請求項３１に記載の情報処理装置。

【請求項３３】　前記プログラム処理手段は、使用中の機能に対する処理要求が行われた場合、前記機能の使用が終了するまで、前記処理要求を一時的に保留することを特徴とする請求項３１または３２に記載の情報処理装置。

【請求項３４】　前記処理要求は、前記プログラムの一部の機能に対する削除要求または置換要求であることを特徴とする請求項３３に記載の情報処理装置。

【請求項３５】　クラスデータから構成される機能の使用状況を監視するステップと、
前記クラスデータから構成される機能の使用状況に基づいて、前記クラスデータに対する処理を実行するかどうかを決定するステップとを備えることを特徴とする情報処理方法。

【請求項３６】　複数の機能からなるプログラムの実行状況を、それぞれの機能ごとに監視するステップと、
前記プログラムの機能の実行状況に基づいて、前記プログラムの機能に対する処理を行うステップとを備えることを特徴とする情報処理方法。

【請求項３７】　前記プログラムの実行中に実行されない一部の機能がある場合、前記実行されない一部の機能の削除または置換を、前記プログラムの実行中に許可することを特徴とする請求項３６に記載の情報処理方法。

【請求項３８】　前記プログラムの実行中に実行が終了した一部の機能がある場合、前記実行が終了した一部の機能の削除または置換を、前記プログラムの実行中に許可することを特徴とする請求項３６に記載の情報処理方法。

【請求項３９】　前記プログラムの実行中の機能または実行予定のある機能に対して削除要求または置換要求があった場合、前記削除要求または前記置換要求があった機能の実行後に、前記機能の削除または置換を行うことを特徴とする請求項３６に記載の情報処理方法。

【請求項４０】　少なくとも第１の機能と第２の機能とを含むプログラムの実行状況を、それぞれの機能ごとに監視するステップと、
前記プログラムの実行中に前記第１の機能の削除要求を行うステップと、
前記プログラムが前記第２の機能を実行している時に、前記第１の機能を前記プログラムから削除するステップとを備えることを特徴とする情報処理方法。

【請求項４１】　少なくとも第１の機能と第２の機能とを含むプログラムの実行状況を、それぞれの機能ごとに監視するステップと、
前記プログラムの実行中に前記第１の機能の置換要求を行うステップと、
前記プログラムが前記第２の機能を実行している時に、前記プログラムの前記第１の機能を置換するステップとを備えることを特徴とする情報処理方法。

【請求項４２】　オブジェクトに追加されたクラスデータを記憶する機能と、

前記クラスデータの使用状況を監視する機能と、
前記クラスデータの使用状況に基づいて、前記クラスデータの処理を行う機能とをコンピュータに実行させるプログラムを格納した前記コンピュータが読み取り可能な記憶媒体。
【請求項４３】　クラスデータの使用状況を監視するエージェントクラスオブジェクトを生成するためのエージェントクラスを格納する機能と、
前記クラスデータに対する処理が行われる場合、前記エージェントクラスに基づいて前記エージェントクラスオブジェクトを生成する機能とをコンピュータに実行させるプログラムを格納した前記コンピュータが読み取り可能な記憶媒体。
【請求項４４】　プログラムを記憶する機能と、
前記プログラムの使用状況を監視する機能と、
前記プログラムの中に使用されていない一部のプログラムコードが存在する場合、前記使用されていない一部のプログラムコードについて処理を行う機能とをコンピュータに実行させるプログラムを格納した前記コンピュータが読み取り可能な記憶媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、情報処理装置に関し、特に、分散オブジェクト指向プログラミングを行う場合などに適用して好適なものである。
【０００２】
【従来の技術】従来、分散オブジェクト指向技術であるCORBA（Common Object Request Broker Architecture）、DCOM（Distributed Component Object Model）、JavaRMI（Java Remote Method Invocation）などを利用することにより、ローカルに存在するオブジェクトに対してメソッドを発行するのとほとんど同じ手続で、遠隔のコンピュータに存在するオブジェクトに対してメソッドを発行することが可能なことから、分散システムを比較的容易に開発することが可能だった。
【０００３】
【発明が解決しようとする課題】しかしながら、従来の分散オブジェクト指向技術では、動作中のサーバオブジェクトの一部のプログラムコードを置き換える機能や、一部のプログラムコードを遠隔実行する機能がないため、分散オブジェクトの運用を柔軟に行うことができなかった。
【０００４】例えば、データベースサーバオブジェクトにおいて、新たな問い合わせコマンドに対応するプログラムを実行する場合には、そのサーバオブジェクトを終了させ、新たに追加するプログラムをサーバオブジェクトが使用できるように設定を変更した後、サーバオブジェクトを再起動しなければならなかった。
【０００５】このため、分散システムにおけるサーバオブジェクトのプログラムコードの追加・更新には一般に長い時間を要することから、更新されるサービスに関連するサーバオブジェクトの数が多くなるにつれて、そのサービスを停止しなければならない時間が長くなるという問題があった。
【０００６】また、同一のオブジェクトに対して複数のメソッド呼出しを行い、それらの結果に依存して次のメソッド呼出しのパラメータを決定する場合や、メソッド呼出しのパラメータや戻り値のデータ量が大きい場合には、メソッド呼出しによるネットワーク遅延の影響が大きく、分散システムの利点が生かされないという問題があった。
【０００７】また、サーバオブジェクトをある程度汎用的な目的に使えるようにしたり、サーバの動作を呼出し側のプログラムでカスタマイズできるようにするためには、サーバオブジェクトのメソッドインタフェースを増やして細分化しなければならなかった。
【０００８】このため、従来の分散システムでは、インタフェースが複雑になって分かりにくいものになるだけでなく、１つの目的を達成するために呼び出さなければならないメソッドの数が増加し、サービスの応答速度がネットワーク遅延により劣化するという問題があった。
【０００９】さらに、従来の分散システムでは、プログラムコードの追加・更新を行う場合、インタフェースのセキュリティを確保するための保守・運用に関わる管理者の負担が大きいという問題があった。
【００１０】そこで、本発明の第１の目的は、動作中のプログラムを停止させることなく、そのプログラムコードの一部を削除したり、置き換えたりすることを可能とすることである。
【００１１】また、本発明の第２の目的は、プログラマにクラス管理の知識がなくても、プログラムを容易にカスタマイズできるようにすることである。また、本発明の第３の目的は、分散システムにおけるプログラムコードの追加・更新のインターフェイスのセキュリティを向上させることである。
【００１２】
【課題を解決するための手段】上述した課題を解決するために、本発明によれば、オブジェクトに追加されたクラスデータを記憶するクラスデータ記憶手段と、クラスデータの使用状況を監視するクラスデータ監視手段と、クラスデータの使用状況に基づいて、クラスデータの処理を行うクラスデータ処理手段とを備えている。
【００１３】このことにより、プログラムをクラスデータで構成することが可能となるとともに、プログラムの使用状況を追加されたクラスデータごとに監視することが可能となり、プログラムが動作している場合にどのクラスデータが使用されているかを判別することが可能と

なることから、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【００１４】また、本発明の一態様によれば、処理要求があったクラスデータが使用中の場合、そのクラスデータの使用が終了するまで、そのクラスデータの処理を延期するようにしている。

【００１５】このことにより、動作中のプログラムに対して削除や置換などの処理要求が行われた場合においても、そのプログラムの動作を停止させることなく、そのプログラムの一部の機能に対する削除や置換などの処理を行わせることが可能となり、プログラムのカスタマイズ時におけるプログラマの負担を軽減することが可能となる。

【００１６】また、本発明の一態様によれば、クラスデータについての処理要求を受け付けるかどうかを検査する検査手段を備えている。このことにより、プログラムコードの追加要求や更新要求を却下することが可能となり、プログラムコードの追加・更新のインターフェイスのセキュリティを向上させることが可能となる。

【００１７】また、本発明の一態様によれば、クラスデータの追加要求単位でクラスデータをクラスデータ記憶手段に追加するようにしている。このことにより、クラスデータの追加要求を識別することが可能となり、追加要求時のクラスデータからなる機能単位でプログラムに対する処理を行うことが可能となることから、プログラマにクラス管理の知識がなくても、プログラムを容易にカスタマイズすることが可能となる。

【００１８】また、本発明の一態様によれば、クラスデータの使用状況を監視するエージェントクラスオブジェクトを生成するためのエージェントクラスを設け、クラスデータに対する処理が行われる場合、エージェントクラスに基づいてエージェントクラスオブジェクトを生成するようにしている。

【００１９】このことにより、監視対象となるプログラム自体にそのプログラムを監視する機能を容易に付加することが可能となり、プログラマにクラス管理の知識がなくても、プログラムの使用状況の監視を容易に行うことが可能となるとともに、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【００２０】また、本発明の一態様によれば、エージェントクラスは、エージェントクラスオブジェクトに追加されたクラスデータの管理を行うクラス管理オブジェクトを、クラスデータの追加要求が行われるごとに生成するようにしている。

【００２１】このことにより、プログラムに新たに追加された機能ごとにプログラムの監視を行うことが可能となり、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となることから、プログラムの一部の機能を特定するだけで、そのプログラムの機能の削除や置換をそのプログラムの動作中に行うことが可能となる。

【００２２】また、本発明の一態様によれば、クラスデータの追加要求に対して生成されたクラス管理オブジェクトとクラス管理オブジェクトの識別情報との対応関係を記憶するようにしている。

【００２３】このことにより、クラス管理オブジェクトに与えられた名前などの識別情報を指定するだけで、プログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となり、プログラムをカスタマイズする際のプログラマの負担を軽減することが可能となる。

【００２４】また、本発明の一態様によれば、クラス管理オブジェクトは、そのクラス管理オブジェクトに固有のクラスデータを格納する局所クラスデータテーブルを備えている。

【００２５】このことにより、他のオブジェクトから送られてきたクラスデータとクラス管理オブジェクトが既に管理しているクラスデータとの間に衝突が発生した場合においても、他のオブジェクトから送られてきたクラスデータをクラス管理オブジェクトが既に管理しているクラスデータと独立に保有することが可能となり、エージェントクラスオブジェクトは、他のオブジェクトから送られてきたクラスデータを遠隔実行することが可能となる。

【００２６】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータの削除要求を記憶する削除フラグを備えている。このことにより、クラスデータに対する削除要求を記憶しておくことが可能となり、削除要求されたクラスデータが使用中の場合においても、そのクラスデータに対する削除要求を受け入れることが可能となることから、クラスデータに対する削除要求が拒否されたために、そのクラスデータに対する削除要求を改めて行う必要がなくなり、プログラマがクラスデータを削除する際の負担を軽減することが可能となる。

【００２７】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータの置換要求時の新たなクラスデータを記憶する置換クラスデータ記憶領域を備えている。

【００２８】このことにより、置換要求時の新たなクラスデータを記憶しておくことが可能となり、置換要求されたクラスデータが使用中の場合においても、そのクラスデータに対する置換要求を受け入れることが可能となることから、クラスデータに対する置換要求が拒否されたために、そのクラスデータに対する置換要求を改めて行う必要がなくなり、プログラマがクラスデータを置換する際の負担を軽減することが可能となる。

【００２９】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータを実行しているスレッ

ドの識別情報を記憶する実行中スレッドリストを備えている。

【００３０】このことにより、実行中スレッドリストを参照するだけで、実行されているクラスデータが存在するかどうかを判別することができ、クラスデータの使用状況を容易に把握することが可能となる。

【００３１】また、本発明の一態様によれば、クラス管理オブジェクトは、追加されたクラスデータに対応するクラス名を記憶するクラス名リストを備えている。このことにより、クラス管理オブジェクトは、自己が管理しているクラスデータを容易に把握することが可能となり、追加されたクラスデータを単位としてクラスデータに対する処理を行うことが可能となる。

【００３２】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する参照オブジェクトリストと、クラスデータの参照元のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する被参照オブジェクトリストとを備える。

【００３３】このことにより、クラスデータの参照関係を考慮しながらクラスデータに対する処理を行うことが可能となり、プログラマがクラスデータの参照関係を認識する必要がなくなることから、プログラムのカスタマイズを容易に行うことが可能となる。

【００３４】また、本発明の一態様によれば、エージェントクラスは、他のエージェントクラスオブジェクトからのアクセス権をチェックするアクセス制御クラスを備えている。

【００３５】このことにより、エージェントクラスオブジェクトは、他のエージェントクラスオブジェクトからのアクセスがあった場合、アクセス制御クラスに基づいてアクセス制御オブジェクトを生成することが可能となることから、エージェントクラスオブジェクトからのアクセスがある度に、アクセス権のチェックを行うことが可能となるとともに、エージェントクラスオブジェクトごとにアクセス条件を容易に変更することが可能となり、プログラムコードの追加・更新のインターフェイスのセキュリティを向上させることが可能となる。

【００３６】また、本発明の一態様によれば、エージェントクラスオブジェクトにクラスデータを追加する外部公開メソッドを備えている。このことにより、プログラムの実行中においても、そのプログラムに対してクラスデータを新たに追加することが可能となる。

【００３７】また、本発明の一態様によれば、エージェントクラスオブジェクトに追加されたクラスデータに対応するクラス管理オブジェクトを生成するクラス管理オブジェクト生成手段と、クラス管理オブジェクト内のクラス名リストにクラスデータのクラス名を登録するクラス名登録手段と、クラスデータを前記共有クラスデータテーブルに書き込むクラスデータテーブル書き込み手段と、クラス管理オブジェクトの識別情報を前記クラス管理オブジェクトテーブルに登録するクラス管理オブジェクトテーブル登録手段とを備えている。

【００３８】このことにより、エージェントクラスオブジェクトは、追加されたクラスデータごとにその使用状況を監視することが可能となるとともに、追加されたクラスデータを容易に特定してそのクラスデータに対する処理を実行することが可能となる。

【００３９】また、本発明の一態様によれば、エージェントクラスオブジェクトに追加されたクラスデータのクラス名が、共有クラスデータテーブルに登録されているクラスデータのクラス名と重複する場合、エージェントクラスオブジェクトに追加されたクラスデータを共有クラスデータテーブルから削除するようにしている。

【００４０】このことにより、共有クラスデータテーブル内で同一のクラスの衝突が発生することを防止することができる。また、本発明の一態様によれば、エージェントクラスオブジェクトに保有されているクラスデータを実行する外部公開メソッドを備えている。

【００４１】このことにより、エージェントクラスオブジェクトはプログラムを実行させることが可能となる。また、本発明の一態様によれば、実行中のクラスデータが他のクラスデータを参照している場合、実行中のクラスデータを管理しているクラス管理オブジェクトに実行情報を設定するとともに、参照先のクラスデータを管理しているクラス管理オブジェクトにも実行情報を設定するようにしている。

【００４２】このことにより、参照元のクラスデータの実行中に参照先のクラスデータの削除要求や置換要求がなされても、参照元のクラスデータの実行が終了するまで参照先のクラスデータの削除や置換を一時的に保留しておくが可能となり、クラスデータの実行が途中で不可能になることを防止することが可能となる。

【００４３】また、本発明の一態様によれば、実行中スレッドリストが空で、かつ、削除フラグがオンのクラス管理オブジェクトが存在している場合、そのクラス管理オブジェクトのクラス名リストのクラス名に対応するクラスデータを共有クラスデータテーブルから削除するようにしている。

【００４４】このことにより、削除要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を削除することが可能となることから、プログラムの一部の機能を削除する際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【００４５】また、本発明の一態様によれば、実行中スレッドリストが空で、かつ、置換クラスデータ記憶領域にクラスデータが記憶されている場合、共有クラスデー

タテーブルに記憶されている置換対象のクラスデータを、置換クラスデータ記憶領域に記憶されているクラスデータで置換するようにしている。

【００４６】このことにより、置換要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を他の機能で置換することが可能となることから、プログラムの一部の機能を置換する際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【００４７】また、本発明の一態様によれば、エージェントクラスオブジェクトからクラスデータを削除する外部公開メソッドを備えている。このことにより、エージェントクラスオブジェクトはプログラムの一部を削除することが可能となる。

【００４８】また、本発明の一態様によれば、クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されている場合、そのスレッドで実行されているクラスデータの削除を中止し、そのクラス管理オブジェクトの削除フラグをオンに設定するようにしている。

【００４９】このことにより、実行中のクラスデータに対して削除要求が行われた場合においても、そのクラスデータの実行を停止させることなく、そのクラスデータに対する削除要求を受け入れることが可能となることから、そのクラスデータの実行に影響を与えることなく、そのクラスデータを削除することが可能となる。

【００５０】また、本発明の一態様によれば、エージェントクラスオブジェクトのクラスデータを置換する外部公開メソッドを備えている。このことにより、エージェントクラスオブジェクトはプログラムの一部を置換することが可能となる。

【００５１】また、本発明の一態様によれば、クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されている場合、そのスレッドで実行されているクラスデータの置換を中止し、そのクラス管理オブジェクトの置換クラスデータ記憶領域に新たなクラスデータを設定するようにしている。

【００５２】このことにより、実行中のクラスデータに対して置換要求が行われた場合においても、そのクラスデータの実行を停止させることなく、そのクラスデータに対する置換要求を受け入れることが可能となり、そのクラスデータの実行に影響を与えることなく、そのクラスデータを置換することが可能となる。

【００５３】また、本発明の一態様によれば、削除要求や置換要求に対し、クラス管理オブジェクトの被参照オブジェクトリストが空でない場合、エラーを通知し、クラス管理オブジェクトの参照オブジェクトリストが空でない場合、参照先のクラス管理オブジェクトの被参照オブジェクトリストから、参照元のクラス管理オブジェクトの識別情報を削除するようにしている。

【００５４】このことにより、他のクラスデータが参照しているクラスデータが消失することを防止することが可能となるとともに、不要となった情報を削除することが可能となり、プログラムをカスタマイズする際の利便性を向上させることが可能となる。

【００５５】また、本発明の一態様によれば、他のエージェントクラスオブジェクトから送られたクラスデータを実行する外部公開メソッドを備えている。このことにより、エージェントクラスオブジェクトはプログラムの遠隔実行を行うことが可能となる。

【００５６】また、本発明の一態様によれば、他のエージェントクラスオブジェクトから送られたクラスデータをクラス管理オブジェクトの局所クラスデータテーブルに格納するようにしている。

【００５７】このことにより、クラス管理オブジェクトに対応して既に格納されているクラスデータとの衝突を回避することが可能となる、エージェントクラスオブジェクトによる遠隔実行を円滑に行うことが可能となる。

【００５８】また、本発明の一態様によれば、クラスデータ及びクラスデータから生成されたオブジェクトのシリアライズデータに基づいて、そのクラスデータから生成されたオブジェクトを復元するようにしている。

【００５９】このことにより、クラスデータから生成されたオブジェクトを他のエージェントクラスオブジェクトに送信して、そのオブジェクトの遠隔実行を行うことが可能となる。

【００６０】また、本発明の一態様によれば、実行対象のクラスデータとともにその実行対象のクラスデータが参照しているクラスデータを、他のエージェントクラスオブジェクトに送信するようにしている。

【００６１】このことにより、参照関係を有しているクラスデータについても、遠隔実行を行うことが可能となる。また、本発明の一態様によれば、送信先のエージェントクラスオブジェクトに存在しているクラスデータを除外して、実行対象のクラスデータをその送信先のエージェントクラスオブジェクトに送信するようにしている。

【００６２】このことにより、遠隔実行を行う際の通信量を減らすことが可能となり、遠隔実行の速度を上げることが可能となる。また、本発明の一態様によれば、複数の機能からなるプログラムの実行状況を、それぞれの機能ごとに監視し、そのプログラムの機能の実行状況に基づいて、そのプログラムの機能に対する処理を行うようにしている。

【００６３】このことにより、プログラムの実行中に一部の機能が実行されない場合や、プログラムの実行中に一部の機能の実行が終了した場合、その一部の機能の削除または置換をそのプログラムの実行中に行うことが可能となり、プログラムのカスタマイズを迅速に行うこと

が可能となる。

【００６４】

【発明の実施の形態】以下、本発明の実施例について図面を参照しながら説明する。図１は、本発明の一実施例に係わる情報処理装置の機能的な構成を示すブロック図である。

【００６５】図１において、クラスデータ記憶手段１は、オブジェクトに追加されたクラスデータを記憶する。クラスデータ監視手段２は、クラスデータ記憶手段１に記憶されているクラスデータの使用状況を追加されたクラスデータごとに監視する。クラスデータ処理手段３は、クラスデータ監視手段２で監視されているクラスデータの使用状況に基づいて、クラスデータの処理を追加されたクラスデータごとに行う。

【００６６】このため、クラスデータ監視手段２は、プログラムに新たに追加された機能ごとにプログラムの監視を行うことが可能となり、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となる。この結果、クラスデータ処理手段３は、プログラムの一部の機能を特定するだけで、そのプログラムの機能に対する処理をそのプログラムの動作中に行うことが可能となる。

【００６７】図２は、図１のクラスデータ処理手段３の構成例を示すブロック図である。図２において、受付手段１１は、クラスデータ記憶手段１に記憶されているクラスデータについての削除要求や置換要求などを受け付ける。判定手段１２は、クラスデータ監視手段２での監視結果に基づいて、削除要求や置換要求などがあったクラスデータが使用中かどうかを判定する。延期手段１３は、クラスデータが使用中の場合、そのクラスデータの使用が終了するまで、そのクラスデータの削除や置換などを延期する。

【００６８】ここで、受付手段１１は、実行中のクラスデータに対して削除要求や置換要求が行われた場合、そのクラスデータの実行を停止させることなく、そのクラスデータに対する削除要求や置換要求を受け付ける。そして、遅延手段１３は、判定手段１２による判定結果を参照することにより、削除対象または置換対象となっているクラスデータの使用が終了したがどうかを識別し、削除対象または置換対象となっているクラスデータの使用が終了してから、受付手段１１が受け付けた削除や置換などの処理を実行する。

【００６９】このことにより、削除要求や置換要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を削除したり、他の機能で置換したりすることが可能となり、プログラムの一部の機能を削除したり置換したりする際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【００７０】図３は、本発明の一実施例に係わるコンピュータの機能的な構成を示すブロック図である。図３において、開発言語のライブラリの基本クラスとしてエージェントクラスが設けられ、このエージェントクラスはエージェントクラスライブラリ３１に格納されている。プログラマは、システム開発を行う場合、この基本クラスとしてのエージェントクラスに基づいて、サブクラスを生成することにより、アプリケーションを記述する。

【００７１】エージェントクラスには、クラスデータの追加要求に対して生成されたクラス管理オブジェクト２４ａ～２４ｍとクラス管理オブジェクト２４ａ～２４ｍの名前との対応関係を記憶したクラス管理オブジェクトテーブル２３、エージェントクラスオブジェクト２２ａ～２２ｎに追加されたクラスデータの管理を行うクラス管理オブジェクト２４ａ～２４ｍと、クラス管理オブジェクト２４ａ～２４ｍで管理されるクラスデータを記憶する共有クラスデータテーブル２５及びクラスデータに対する処理を実行する外部公開メソッド２６が設けられている。

【００７２】エージェントクラスオブジェクト２２ａ～２２ｎは、エージェントクラスライブラリ３１に格納されているエージェントクラスのインスタンスとして、コンピュータ２１上に生成される。エージェントクラスオブジェクト２２ａのクラス管理オブジェクトテーブル２３には、例えば、クラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”がそれぞれ登録されるとともに、それぞれの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”に対応するクラス管理オブジェクト２４ａ～２４ｍへのポインタが格納されている。

【００７３】エージェントクラスオブジェクト２２ａの外部公開メソッド２６として、例えば、ｒｅｍｏｖｅ（）メソッド２６ａ、ａｄｄ（）メソッド２６ｂ、ｇｅｔ（）メソッド２６ｃ、ｓｅｔ（）メソッド２６ｄ、ｅｖａｌ（）メソッド２６ｅ、ｃａｌｌ（）メソッド２６ｆが設けられている。

【００７４】ｒｅｍｏｖｅ（）メソッド２６ａは、削除対象となるクラスデータを管理しているクラス管理オブジェクト２４ａ～２４ｍの名前を引数として、共有クラスデータテーブル２５からクラスデータを削除する。

【００７５】ａｄｄ（）メソッド２６ｂは、追加対象となるクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前及び追加対象となるクラスデータを引数として、追加対象となるクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍを新たに生成するとともに、そのクラス管理オブジェクト２４ａ～２４ｍの名前をクラス管理オブジェクトテーブル２３に登録し、追加対象となるクラスデータを共有クラスデータテーブル２５に格納する。

【００７６】ｇｅｔ（）メソッド２６ｃは、取得対象と

なるクラスデータを管理しているクラス管理オブジェクト２４ａ～２４ｍの名前を引数として、共有クラスデータテーブル２５からクラスデータをコピーする。

【００７７】ｓｅｔ（）メソッド２６ｄは、置換対象となるクラスデータを管理しているクラス管理オブジェクト２４ａ～２４ｍの名前及び置換する新たなクラスデータを引数として、共有クラスデータテーブル２５の置換対象となるクラスデータを新たなクラスデータで置換する。

【００７８】ｅｖａｌ（）メソッド２６ｅは、実行対象となるクラスデータ及びそのクラスデータのオブジェクトのシリアライズデータを引数として、そのクラスデータまたはそのオブジェクトについての遠隔実行を行う。

【００７９】ｃａｌｌ（）メソッド２６ｆは、実行対象となるクラスデータを管理しているクラス管理オブジェクト２４ａ～２４ｍの名前を引数として、共有クラスデータテーブル２５に格納されているクラスデータからオブジェクトを生成して実行する。

【００８０】コンピュータ２１には、また、通信デーモン２８、ネットワークソケットＡＰＩ（アプリケーション　プログラミング　インターフェイス）２９、オペレーティング・システム３０及びシステムクラスライブラリ３２が設けられている。

【００８１】外部公開メソッド２６が呼ばれた際には、通信スレッド２７ａ～２７ｋが割り当てられ、１つのプログラム中で複数の実行の流れ（例えば、メソッドや関数）が、データ領域を共有しながら並列して行われることが可能となっている。なお、このマルチスレッド機能は、Ｊａｖａ言語などでは標準的に備わっているものである。

【００８２】コンピュータ２１内のエージェントクラスオブジェクト２２ａ～２２ｎ間での情報の通信や、他のコンピュータとの間で情報の通信は、ネットワークソケットＡＰＩを介して行われる。

【００８３】エージェントクラスオブジェクト２２ａ～２２ｎのクラス置換機能や遠隔実行機能が実際のシステムで利用される状況としては、例えば、ソフトウェアの遠隔保守が挙げることができる。そして、こられのクラス置換機能や遠隔実行機能を利用することにより、サーバを停止させることなく、ソフトウェアの入れ替えを遠隔に行うことが可能になるとともに、複数のサーバをバッチ処理的に行うことができるようになる。

【００８４】このため、サーバソフトの一部だけを対象として、プログラムを変更する場合でも、サーバを停止し、プログラムを変更した後に、サーバを再起動するような手間をなくすことが可能となり、特に、分散システムの場合には、管理者の負担を大幅に軽減することが可能となる。また、一般的なアプリケーションについても、エージェントクラスオブジェクト２２ａ～２２ｎ内部で動作するクラスとして作成することにより、アプリケーションのカスタマイズをより柔軟に行うことが可能となる。

【００８５】このように、図３の実施例では、複数のエージェントクラスオブジェクト２２ａ～２２ｎが連携し、動作中のエージェントクラスオブジェクト２２ａ～２２ｎが、そのエージェントクラスオブジェクト２２ａ～２２ｎを構成するプログラムコードであるクラスデータの追加要求または削除要求を受け付ける場合に、それぞれのエージェントクラスオブジェクト２２ａ～２２ｎ内に共有クラスデータテーブル２５を設け、追加要求の発生ごとにクラス管理オブジェクト２４ａ～２４ｍを生成する。

【００８６】そして、そのクラス管理オブジェクト２４ａ～２４ｍが、共有クラスデータテーブル２５の検索や追加されたクラス群の実行及び削除を管理することにより、追加されたクラス群を共有することが可能となるとともに、部分的なクラス群を動的に削除・更新することが可能となる。

【００８７】また、ランタイム環境におけるプログラムの置換機能やプログラムの一部の遠隔実行を利用することにより、分散型ネットワーク管理システム、分散型ネットワークサービス、分散型グループウェアなどでの柔軟な相互連携や動的な負荷分散が可能となる。

【００８８】さらに、動作中のサーバオブジェクトのプログラムコードの追加・更新をサーバオブジェクトの動作中に行うことが可能となり、更新されるサービスに関連するサーバオブジェクトを完全に停止させる必要がなくなることから、サービスの停止期間を飛躍的に短くすることが可能となる。

【００８９】なお、オブジェクトを構成するクラスデータを監視する機能をそのオブジェクトに設けたものをエージェントと呼ぶ。図４は、図３のクラス管理オブジェクトの構成例を示す図である。

【００９０】図４において、クラス管理オブジェクト２４ａ～２４ｍには、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）４１、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）４２、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）４３、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）４４、クラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）４５、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）４６、被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）４７及びクラスデータの検索を行うクラス探索機能４８が設けられている。

【００９１】局所クラスデータテーブル４１は、プログラムの遠隔実行時に受信したクラスデータを保存するもので、遠隔実行時に他のエージェントクラスオブジェクト２２ａ～２２ｎからクラスデータが送られてきた場合、そのクラスデータをクラス管理オブジェクト２４ａ～２４ｍにより既に管理されているクラスデータと独立

に保有することにより、送信されたクラスデータとクラス管理オブジェクト２４ａ～２４ｍが既に管理しているクラスデータとの間の衝突を防止することを可能とし、クラスデータの遠隔実行を行うことを可能とするものである。

【００９２】削除フラグ４２は、共有クラスデータテーブル２５に追加されたクラスデータの削除要求が発生したかどうかを示すもので、クラスデータに対する削除要求を削除フラグ４２に記憶しておくことにより削除要求されたクラスデータが使用中の場合においても、そのクラスデータに対する削除要求を受け入れることを可能とするものである。

【００９３】置換クラスデータ記憶領域４３は、共有クラスデータテーブル２５に追加されたクラスデータの置換要求が発生した場合に、置換する新しいクラスデータを一時的に保存しておくもので、置換要求時の新たなクラスデータを置換クラスデータ記憶領域４３に記憶しておくことにより、置換要求されたクラスデータが使用中の場合においても、そのクラスデータに対する置換要求を受け入れることを可能とするものである。

【００９４】実行中スレッドリスト４４は、共有クラスデータテーブル２５に追加されたクラスデータを実行しているスレッドＩＤを記憶するもので、実行中スレッドリスト４４にスレッドＩＤを記憶することにより、実行中スレッドリスト４４を参照するだけで、実行されているクラスデータが存在するかどうかを判別することを可能とするものである。

【００９５】クラス名リスト４５は、共有クラスデータテーブル２５に追加されたクラスデータの各クラス名を記憶するもので、クラス管理オブジェクト２４ａ～２４ｍが管理しているクラスデータを把握することを可能とし、クラス管理オブジェクト２４ａ～２４ｍが管理するクラスデータを単位として処理を行うことを可能とするものである。

【００９６】参照オブジェクトリスト４６は、共有クラスデータテーブル２５に追加されたクラスデータが利用する他のクラスデータを、どのクラス管理オブジェクト２４ａ～２４ｍが管理しているかを記憶するもので、参照先のクラスデータを考慮しながらクラスデータに対する処理を行うことを可能とするものである。

【００９７】被参照オブジェクトリスト４７は、クラスデータを利用している他のクラスデータを、どのクラス管理オブジェクト２４ａ～２４ｍが管理しているかを記憶するもので、参照元のクラスデータを考慮しながらクラスデータに対する処理を行うことを可能とするものである。

【００９８】図５は、エージェントクラスの構成例を示す図である。エージェントクラスは、クラスライブラリ５１の基本クラス５２として設けられ、この基本クラス５２には、クラス管理オブジェクトテーブル５３、クラス管理オブジェクト５、共有クラスデータテーブル５５及び外部公開メソッド５６が設けられている。また、クラスライブラリ５１には、基本クラス５２から生成されたサブクラス５７が登録され、エージェントクラスオブジェクト２２ａ～２２ｎは、サブクラス５７からのインスタンス５８ａ～５８ｊとして生成することができる。

【００９９】このように、分散システムの開発者は、エージェントクラスのサブクラス５７を作成して、アプリケーションを記述することにより、クラス管理に関する特別な知識がなくとも、ランタイム環境でのクラスデータの置換機能を実現したり、外部のエージェントクラスオブジェクト２２ａ～２２ｎからクラスデータを送り付けることにより、遠隔実行機能を利用した柔軟なシステムを構築することができる。

【０１００】図６は、クラスデータ追加時のクラス管理オブジェクトの生成方法を示す図である。図６において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣをエージェントクラスオブジェクト３０１に追加する場合、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル３０２に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト３０３が生成される。

【０１０１】クラス管理オブジェクト３０３には、クラス管理オブジェクト３０３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名がそれぞれ登録されるとともに、クラス管理オブジェクト３０３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブルに登録される。

【０１０２】クラスデータＡ、クラスデータＢ及びクラスデータＣがエージェントクラスオブジェクト３０１に追加された後、“ｆｕｎｃｔｉｏｎ－２”という名前でクラスデータＤ及びクラスデータＥをエージェントクラスオブジェクト３０１に追加する場合、クラスデータＤ及びクラスデータＥが共有クラスデータテーブル３０２に格納されるとともに、クラスデータＤ及びクラスデータＥを管理するクラス管理オブジェクト３０４が新たに生成される。

【０１０３】クラス管理オブジェクト３０４には、クラス管理オブジェクト３０４が管理するクラスデータＤ及びクラスデータＥのクラス名がそれぞれ登録されるとともに、クラス管理オブジェクト３０４の名前として、“ｆｕｎｃｔｉｏｎ－２”がクラス管理オブジェクトテーブルに登録される。

【０１０４】クラスデータＤ及びクラスデータＥがエージェントクラスオブジェクト３０１に追加された後、“ｆｕｎｃｔｉｏｎ－３”という名前でクラスデータＦ、クラスデータＧ及びクラスデータＨをエージェントクラスオブジェクト３０１に追加する場合、クラスデー

タＦ、クラスデータＧ及びクラスデータＨが共有クラスデータテーブル３０２に格納されるとともに、クラスデータＦ、クラスデータＧ及びクラスデータＨを管理するクラス管理オブジェクト３０５が新たに生成される。

【０１０５】クラス管理オブジェクト３０５には、クラス管理オブジェクト３０５が管理するクラスデータＦ、クラスデータＧ及びクラスデータＨのクラス名がそれぞれ登録されるとともに、クラス管理オブジェクト３０５の名前として、“ｆｕｎｃｔｉｏｎ－３”がクラス管理オブジェクトテーブルに登録される。

【０１０６】このように、クラスデータにより構成される機能が、エージェントクラスオブジェクト３０１に追加されるごとに、クラス管理オブジェクト３０３～３０５を生成することにより、クラスデータにより構成される機能ごとに、プログラムの監視を行うことが可能となる。このため、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となり、そのプログラムの一部の機能の削除や置換をそのプログラムの動作中に行うことが可能となる。

【０１０７】また、クラス管理オブジェクト３０３～３０５の名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－３”をクラス管理オブジェクトテーブルに登録しておくことにより、クラス管理オブジェクト３０３～３０５の名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－３”を指定するだけで、プログラマがプログラムに追加した機能を特定することが可能となり、プログラムをカスタマイズする際の利便性を向上させることが可能となる。

【０１０８】次に、本発明の一実施例に係わるアプリケーションのカスタマイズ方法について説明する。図７は、テンプレートのコード化の例を示す図である。

【０１０９】図７において、例えば、Ｅメールの着信通知をスクリーニングするアプリケーションが、初期設定ファイルを使ったテンプレート（型枠）形式で提供されているものとする。このテンプレートは、特定の発信者からのメールの着信通知にスクリーニングをかけるためのもので、初期設定メニュー８１のフィールド欄８３のホップアップメニュー８２から“Ｆｒｏｍ”を選択し、スクリーニングをかける発信者の名称をキーボードから入力することにより、発信者の名称を内容欄８４に設定するようになっている。

【０１１０】一方、Ｅメール本文の中からキーワードを検索してスクリーニングを行う場合、図７のテンプレートにより提供される機能を用いただけでは、このキーワード検索を行うことは不可能である。このため、Ｅメール自体のテキストをキーワード検索するプログラムコードを外部から持ってくる。そして、図７のテンプレートにより提供されるＥメールの着信通知の部分のプログラムコードを、Ｅメール自体のテキストをキーワード検索するプログラムコードと置換する。

【０１１１】ここで、このプログラムコードの置換を行う際には、クラス管理オブジェクト２４ａ～２４ｍが、Ｅメールの着信通知の部分のプログラムコードの使用状況を監視することにより、Ｅメールの着信通知の部分のプログラムコードが実行されていない時に、プログラムコードの置換を行わせることが可能となる。このため、図７のテンプレートをカスタマイズする際に、図７のテンプレート形式によるプログラムの動作を停止させることなく、図７のテンプレート形式によるプログラムの機能の一部を置換することが可能となる。

【０１１２】このように、テンプレートをコード化するとともに、プログラムコードを機能ごとに監視することにより、カスタマイズするプログラムを動作させながら、構築するアプリケーションに必要なプログラムコードを検索して置換することが可能となり、手元に備わっているテンプレートに制約されることなく、アプリケーションのカスタマイズを柔軟に行うことが可能となる。

【０１１３】図８は、図３のコンピュータのシステム構成例を示す図である。図８において、ＣＰＵ６１、ＲＡＭ６２、ＲＯＭ６３、通信インターフェイス６４、プリンタ６５、ディスプレイ６６、キーボード６７、マウス６８及びドライバ６９が互いに接続され、ドライバ６９には、例えば、ハードディスク７０、フロッピーディスク７１、磁気テープ７２、ＣＤ－ＲＯＭやＤＶＤ－ＲＯＭなどの光ディスク７３、ＩＣメモリカード７４が接続されている。

【０１１４】エージェントクラスは、ＲＯＭ６３、ハードディスク７０、フロッピーディスク７１、磁気テープ７２、ＣＤ－ＲＯＭやＤＶＤ－ＲＯＭなどの光ディスク７３、ＩＣメモリカード７４などに格納され、エージェントクラスからエージェントクラスオブジェクト２２ａ～２２ｎが起動されると、ＣＰＵ６１の制御により、エージェントクラスオブジェクト２２ａ～２２ｎがＲＡＭ６２上に生成される。ここで、エージェントクラスオブジェクト２２ａ～２２ｎに対する操作は、ディスプレイ６６に表示されるテンプレート形式での画面操作やキーボード入力により行うことができる。

【０１１５】なお、プログラムを通信インターフェイス６４を介して送受信することも可能であり、通信インターフェイス６４と接続される通信ネットワークとして、例えば、ＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）、ＷＡＮ（Ｗｉｄｅ Ａｒｅａ Ｎｅｔｗｏｒｋ）、インターネット、アナログ電話網、デジタル電話網（ＩＳＤＮ：Ｉｎｔｅｇｒａｌ Ｓｅｒｖｉｃｅ Ｄｉｇｉｔａｌ Ｎｅｔｗｏｒｋ）、ＰＨＳ（パーソナルハンディシステム）などの無線通信網などを用いることができる。

【０１１６】図９は、図３の通信デーモン２８の動作を示すフローチャートである。図９において、メッセージ受信待ちの状態で（ステップＳ１）、通信デーモン２８

が、他のコンピュータからのメッセージをネットワークソケットＡＰＩ２９を介して受信すると、転送先のエージェントクラスオブジェクト２２ａ～２２ｎの名前“ＡＧＥＮＴ－１”～“ＡＧＥＮＴ－ｎ”、メソッド名及び引数をこのメッセージから取得し（ステップＳ２）、通信スレッド２７ａ～２７ｋを生成する（ステップＳ３）。そして、他のコンピュータからの取得したデータを通信スレッド２７ａ～２７ｋに渡す。

【０１１７】図１０は、図３の通信スレッド２７ａ～２７ｋの動作を示すフローチャートである。図１０において、通信スレッド２７ａ～２７ｋが通信デーモン２８から生成されると（ステップＳ１１）、通信デーモン２８により指定されたエージェントクラスオブジェクト２２ａ～２２ｎの名前“ＡＧＥＮＴ－１”～“ＡＧＥＮＴ－ｎ”から、エージェントクラスオブジェクト２２ａ～２２ｎを特定する（ステップＳ１２）。

【０１１８】次に、通信スレッド２７ａ～２７ｋは、特定したエージェントクラスオブジェクト２２ａ～２２ｎに対し、通信デーモン２８から渡された引数を設定して外部公開メソッド２６の呼び出しを行い（ステップＳ１３）、エージェントクラスオブジェクト２２ａ～２２ｎから戻り値が返されるのを待つ（ステップＳ１４）。

【０１１９】次に、エージェントクラスオブジェクト２２ａ～２２ｎから外部公開メソッド２６の戻り値が返されたら、ネットワーク・ソケットＡＰＩ２９を制御して、外部公開メソッド２６を発行した他のコンピュータのエージェントクラスオブジェクトに戻り値を含むメッセージを返送し（ステップＳ１５）、通信スレッド２７ａ～２７ｋを消滅させる（ステップＳ１６）。

【０１２０】次に、エージェントクラスオブジェクト２２ａ～２２ｎへのクラスデータの追加方法について説明する。図１１は、クラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。

【０１２１】図１１において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能をエージェントクラスオブジェクト１０１に追加する場合、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル１０４に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト１０３が生成され、クラス管理オブジェクト１０３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１０２に登録される。そして、クラス管理オブジェクト１０３の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１０３を特定するポインタがクラス管理オブジェクトテーブル１０２に生成される。

【０１２２】また、クラス管理オブジェクト１０３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１０３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされる。

【０１２３】図１２は、参照関係のあるクラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。図１２において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト１１１に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル１１２に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト１１３が生成されている。

【０１２４】このクラス管理オブジェクト１１３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１１３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）及び参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）が空（ｎｕｌｌ）とされている。

【０１２５】また、クラス管理オブジェクト１１３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１１２に登録され、クラス管理オブジェクト１１３の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１１３を特定するポインタがクラス管理オブジェクトテーブル１１２に生成されている。

【０１２６】この状態で、“ｆｕｎｃｔｉｏｎ－３”という名前でクラスデータＦ及びクラスデータＧからなる機能をエージェントクラスオブジェクト１１１に追加する場合、クラスデータＦ及びクラスデータＧが共有クラスデータテーブル１１５に追加される。

【０１２７】また、クラスデータＦ及びクラスデータＧを管理するクラス管理オブジェクト１１４が新たに生成され、クラス管理オブジェクト１１４の名前として、“ｆｕｎｃｔｉｏｎ－３”がクラス管理オブジェクトテーブル１１２に登録され、クラス管理オブジェクト１１４の名前“ｆｕｎｃｔｉｏｎ－３”に対応して、クラス管理オブジェクト１１４を特定するポインタがクラス管理オブジェクトテーブル１１２に新たに生成される。

【０１２８】さらに、クラス管理オブジェクト１１４のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１１４が管理するクラスデータＦ及びクラスデータＧのクラス名“Ｆ”、“Ｇ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされる。

【０１２９】ここで、クラスデータＦが、例えば、

```
classF０ {
        :
        new Ａ０;
        :
        new Ｇ０;
}
```

のように、共有クラスデータテーブル１１５に既に追加されているクラスデータＡの存在を前提とする場合、クラス管理オブジェクト１１４の参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）には、参照先のクラスデータＡを管理しているクラス管理オブジェクト１１３の参照情報が格納され、クラス管理オブジェクト１１３の被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）には、参照元のクラスデータＦを管理しているクラス管理オブジェクト１１４の被参照情報が格納される。

【０１３０】このため、名前“ｆｕｎｃｔｉｏｎ－１”を引数として、ｒｅｍｏｖｅ（）メソッド２６ａやｓｅｔ（）メソッド２６ｄが呼び出され、クラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能の削除要求や置換要求がエージェントクラスオブジェクト１１１に対して行われた場合でも、クラス管理オブジェクト１１３は、被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）を調べることにより、クラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能の削除要求や置換要求を保留することが可能となり、クラスデータＡを参照しているクラスデータＦの機能が損なわれることを防止することが可能となる。

【０１３１】図１３は、重複するクラスデータのエージェントクラスオブジェクト２２ａ～２２ｎへの登録を拒否する方法を示す図である。図１３において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト１２１に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル１２２に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト１２３が生成されている。

【０１３２】このクラス管理オブジェクト１２３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１２３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０１３３】また、クラス管理オブジェクト１２３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１２２に登録され、クラス管理オブジェクト１２３の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１２３を特定するポインタがクラス管理オブジェクトテーブル１２２に生成されている。

【０１３４】この状態で、名前“ｆｕｎｃｔｉｏｎ－６”、クラスデータＤ、クラスデータＥ及びクラスデータＦを引数として、エージェントクラスオブジェクト１２１のａｄｄ（）メソッド１２５が呼ばれると、クラスデータＤがエージェントクラスオブジェクト１２１の共有クラスデータテーブル１２４に格納されていないことを確認してから、共有クラスデータテーブル１２４にクラスデータＤを新たに格納する。

【０１３５】次に、クラスデータＥが共有クラスデータテーブル１２４に格納されていないことを確認してから、共有クラスデータテーブル１２４にクラスデータＥを新たに格納する。

【０１３６】次に、共有クラスデータテーブル１２４にクラスデータＡを格納する場合、クラスデータＡは共有クラスデータテーブル１２４に既に格納されていることから、クラスデータＤ及びクラスデータＥを共有クラスデータテーブル１２４から削除し、名前“ｆｕｎｃｔｉｏｎ－６”のクラスデータＤ、クラスデータＥ及びクラスデータＡについては、共有クラスデータテーブル１２４に格納しないこととする。

【０１３７】このように、エージェントクラスオブジェクト１２１が、複数のクラスデータを同時に追加したり、置換したりする要求を受ける時に、追加対象の一部のクラス名とエージェントクラスオブジェクト１２１内部に既に存在するクラス名との衝突がクラスデータの追加処理の途中で発生した場合、共有クラスデータテーブル１２４の追加対象のクラスデータを全て削除することにより、共有クラスデータテーブル１２４の元の状態を復元するようにしている。

【０１３８】図１４は、クラスデータ追加処理を示すフローチャートである。ここで、エージェントクラスオブ

ジェクト２２ａ～２２ｎがクラスデータ追加処理を行う場合、ａｄｄ（）メソッド２６ｂが呼び出される。

【０１３９】図１４において、まず、エージェントクラスオブジェクト２２ａ～２２ｎに対するアクセス権のチェックが行われ（ステップＳ２１）、アクセス権がなければ処理が中断される。

【０１４０】次に、クラスデータの追加時に付ける名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”及び追加するクラスデータをａｄｄ（）メソッド２６ｂの引数から取得し（ステップＳ２２）、クラスデータの追加時に付ける名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”をクラス管理オブジェクトテーブル２３から検索する（ステップＳ２３）。

【０１４１】次に、クラスデータの追加時に付ける名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に既に存在するかどうかを調べ（ステップＳ２４）、クラスデータの追加時に付ける名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”がクラス管理オブジェクトテーブル２３に既に存在する場合、エラーを通知して処理を中断する（ステップＳ２５）。

【０１４２】一方、クラスデータの追加時に付ける名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”がクラス管理オブジェクトテーブル２３に存在しない場合、追加要求のあったクラスデータを共有クラスデータテーブル１２５に書き込む（ステップＳ２６）。

【０１４３】次に、同一のクラス名が既に共有クラスデータテーブル２５に存在しているかどうかを判断し（ステップＳ２７）、クラスデータの書込みの途中で、同一のクラス名が既に共有クラスデータテーブル２５に存在していた場合、共有クラスデータテーブル２５に格納した追加要求時のクラスデータを削除し、共有クラスデータテーブル２５を元の状態に戻すとともに（ステップＳ２８）、エラーを通知して処理を終了する（ステップＳ２９）。

【０１４４】一方、クラスデータの書込みの途中で、同一のクラス名が既に共有クラスデータテーブル２５に存在しなかったために、追加要求のあった全てのクラスデータの追加に成功した場合、クラス管理オブジェクト２４ａ～２４ｍを新たに生成する。そして、ａｄｄ（）メソッド２６ｂの引数に設定されていたクラスデータの追加時に付ける名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”をクラス管理オブジェクトテーブル２３に登録し、新たに生成されたクラス管理オブジェクト２４ａ～２４ｍをクラス管理オブジェクトテーブル２３に登録された名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”で識別できるようにする（ステップＳ３０）。

【０１４５】次に、共有クラスデータテーブル２５に追加した全てのクラスデータのクラス名を、新たに生成されたクラス管理オブジェクト２４ａ～２４ｍのクラス名リスト４５に設定する。（ステップＳ３１）。

【０１４６】次に、共有クラスデータテーブル２５に追加したクラスデータに関連するクラスを調べ、新たに生成されたクラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６に参照先情報を設定するとともに（ステップＳ３２）、参照先のクラス管理オブジェクト２４ａ～２４ｍの被参照オブジェクトリスト４７に参照元情報を設定して（ステップＳ３３）、処理を終了する。

【０１４７】次に、本発明の一実施例に係わるアクセス権の制御方法について説明する。クラス置換時のアクセス権チェック用の処理を、例えば、“ｃｈｅｃｋＳｅｔ”という名前で予約し、以下のようなクラスを、“ｃｈｅｃｋＳｅｔ”という名前でエージェントクラスオブジェクト２２ａ～２２ｎに追加するものとする。

【０１４８】

```
class CheckSet {
    void start (RemoteAgent
    requester,
    String name,
    Ogject val)
    throws AccessException {
       if (!requester. get ("name").
       equals ("Admin"))
         throw new AccessException () ;
    }
}
```

そして、クラスデータの登録前に要求元のユーザ名、ホスト名、エージェントクラスオブジェクト２２ａ～２２ｎの名前“ＡＧＥＮＴ－１”～“ＡＧＥＮＴ－ｎ”などからアクセス制御条件をチェックし、クラスデータに対する処理要求を受け付けるかどうかを決定する。すなわち、動作中のエージェントクラスオブジェクト２２ａ～２２ｎが、クラスデータの削除要求または置換要求を受け付ける場合に、名前“ｃｈｅｃｋＳｅｔ”のクラスからオブジェクトを生成して実行する。

【０１４９】この場合、クラスデータ置換要求の要求元

のエージェントの名前が“Ａｄｍｉｎ　”に一致しなければ、置換処理は行われずに、例外が発生する。このアクセス制御条件を記述したクラスは、エージェントクラスオブジェクト２２ａ～２２ｎ内の他のクラスと扱いが同じであるため、運用中のアクセス制御条件の変更が可能となる。すなわち、アクセス制御条件は、エージェントクラスオブジェクト２２ａ～２２ｎ内の置換可能なクラスデータとして管理され、動作中のエージェントクラスオブジェクト２２ａ～２２ｎに対してアクセス制御条件を変更することが可能となる。

【０１５０】このため、プログラムの追加・更新を遠隔のコンピュータから行う場合においても、プログラムコードの追加・更新のインタフェースのセキュリティを確保することが可能となり、分散オブジェクトの保守・運用に関わる管理者の負担を軽減することが可能となる。

【０１５１】図１５は、アクセス権チェック処理を示すフローチャートである。なお、このアクセス権チェック処理は、図１３のエージェントクラスオブジェクト２２ａ～２２ｎに対するクラスデータの追加処理、図１６のクラスデータの取得処理、図１９のクラスデータの実行処理、図２１のクラスデータの削除処理、図２４及び図２５のクラスデータの置換処理、図２８のクラスデータの遠隔実行処理で呼び出される。

【０１５２】図１５において、まず、アクセス権チェックが開始されると（ステップＳ４１）、アクセス制御条件を記述したアクセス権チェックのクラスを共有クラスデータテーブル２５から検索する（ステップＳ４２）。そして、共有クラスデータテーブル２５内にアクセス権チェックのクラスが存在するかどうかを調べ（ステップＳ４３）、共有クラスデータテーブル２５内にアクセス権チェックのクラスが存在する場合、アクセス権チェックのクラスからオブジェクトを生成し、実行する（ステップＳ４４）。

【０１５３】ここで、アクセス権チェックのクラスは、処理の種類と引数、処理要求元のエージェントクラスオブジェクト２２ａ～２２ｎの名前“ＡＧＥＮＴ－１”～“ＡＧＥＮＴ－ｎ”、処理要求元のユーザ名などを元にルールを記述したプログラムであり、エージェントクラスオブジェクト２２ａ～２２ｎに追加される他のクラスデータと同様、エージェントクラスオブジェクト２２ａ～２２ｎの運用中にアクセス権チェックのクラスを置き換えることを許容するものである。

【０１５４】このため、エージェントクラスオブジェクト２２ａ～２２ｎごとにアクセス制御条件を容易に変更することが可能となり、プログラムコードの追加・更新のインタフェースのセキュリティを向上させることが可能となる。

【０１５５】一方、共有クラスデータテーブル２５内にアクセス権チェックのクラスが存在しない場合、デフォルトの条件でアクセス権のチェックを行う（ステップＳ４５）。

【０１５６】次に、アクセス権があるかどうかを判断し（ステップＳ４６）、アクセス権がない場合は、エラーを通知して終了し（ステップＳ４７）、アクセス権がある場合は、エージェントクラスオブジェクト２２ａ～２２ｎに要求があった処理を受け入れる（ステップＳ４８）。

【０１５７】次に、エージェントクラスオブジェクト２２ａ～２２ｎからのクラスデータの取得方法について説明する。図１６は、クラスデータ取得処理を示すフローチャートである。

【０１５８】ここで、エージェントクラスオブジェクト２２ａ～２２ｎがクラスデータ取得処理を行う場合、ｇｅｔ（）メソッド２６ｃが呼び出される。図１６において、まず、エージェントクラスオブジェクト２２ａ～２２ｎに対するアクセス権のチェックが行われ（ステップＳ５１）、アクセス権がなければ処理が中断される。

【０１５９】次に、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”を、ｇｅｔ（）メソッド２６ｃの引数から取得し（ステップＳ５２）、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”を、クラス管理オブジェクトテーブル２３から検索する（ステップＳ５３）。

【０１６０】次に、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されているどうかを調べ（ステップＳ５４）、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されていない場合、エラーを通知して処理を中断する（ステップＳ５５）。一方、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されている場合、そのクラス管理オブジェクト２４ａ～２４ｍのクラス名リスト４５からクラス名を取得し、そのクラス名のクラスデータを共有クラスデータテーブル２５からコピーして（ステップＳ５６）、そのクラスデータを戻り値として要求先に返す（ステップＳ５７）。

【０１６１】次に、エージェントクラスオブジェクト２２ａ～２２ｎでのクラスデータの実行方法について説明する。図１７は、クラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。

【０１６２】図１７において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラ

スオブジェクト１３１に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル１３５に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト１３３が生成されている。

【０１６３】このクラス管理オブジェクト１３３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１３３が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０１６４】また、クラス管理オブジェクト１３３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１３２に登録され、クラス管理オブジェクト１３２の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１３３を特定するポインタがクラス管理オブジェクトテーブル１３２に生成されている。

【０１６５】エージェントクラスオブジェクト１３１に対し、名前“ｆｕｎｃｔｉｏｎ－１”を指定してクラスデータA、クラスデータB及びクラスデータCの実行要求が行われると、クラスデータA、クラスデータB及びクラスデータCからオブジェクトA、オブジェクトB及びオブジェクトCが生成され、スレッド１３４が割り当てられる。

【０１６６】実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）には、オブジェクトA、オブジェクトB及びオブジェクトCに割り当てられたスレッド１３４の識別情報が格納され、オブジェクトA、オブジェクトB及びオブジェクトCの実行が終了した時に、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）から、オブジェクトA、オブジェクトB及びオブジェクトCに割り当てられたスレッド１３４の識別情報が削除される。

【０１６７】このため、クラス管理オブジェクト１３３は、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）を参照することにより、自己が管理するクラスデータA、クラスデータB及びクラスデータCが使用中かどうかを容易に判定することができる。

【０１６８】図１８は、参照関係があるクラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。図１８において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト１４１に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル１４７に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト１４３が生成されている。

【０１６９】このクラス管理オブジェクト１４３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１４３が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）及び参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）が空（ｎｕｌｌ）とされている。

【０１７０】また、クラス管理オブジェクト１４３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１４２に登録され、クラス管理オブジェクト１４３の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１４３を特定するポインタがクラス管理オブジェクトテーブル１４２に生成されている。

【０１７１】さらに、“ｆｕｎｃｔｉｏｎ－４”という名前でクラスデータDからなる機能がエージェントクラスオブジェクト１４１に追加され、クラスデータDが共有クラスデータテーブル１４７に格納されるとともに、クラスデータDを管理するクラス管理オブジェクト１４５が生成されている。

【０１７２】このクラス管理オブジェクト１４５のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１４５が管理するクラスデータDのクラス名“D”が登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０１７３】また、クラス管理オブジェクト１４５の名前として、“ｆｕｎｃｔｉｏｎ－４”がクラス管理オブジェクトテーブル１４２に登録され、クラス管理オブジェクト１４５の名前“ｆｕｎｃｔｉｏｎ－４”に対応して、クラス管理オブジェクト１４５を特定するポインタがクラス管理オブジェクトテーブル１４２に生成されている。

【０１７４】ここで、クラスデータDが、

```
class D0 {
        :
        new A0
        :
        new B0
}
```

のように、例えば、クラスデータA及びクラスデータBの存在を前提としている場合、クラス管理オブジェクト１４５の参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）には、参照先のクラスデータA及びクラスデータBを管理しているクラス管理オブジェクト１４３の参照情報が格納され、クラス管理オブジェクト１４３の被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）には、参照元のクラスデータDを管理しているクラス管理オブジェクト１４５の被参照情報が格納される。この状態で、エージェントクラスオブジェクト１４１に対し、名前“ｆｕｎｃｔｉｏｎ－４”を指定してクラスデータDの実行要求が行われると、クラスデータDからオブジェクトDが生成され、スレッド１４４が割り当てられる。

【０１７５】クラス管理オブジェクト１４５の実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇｔｈｒｅａｄｓ）には、オブジェクトDに割り当てられたスレッド１４４の識別情報が格納されるとともに、クラスデータDが参照しているクラスデータA及びクラスデータBを管理しているクラス管理オブジェクト１４３の実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）に対しても、オブジェクトDに割り当てられたスレッド１４４の識別情報が格納される。

【０１７６】そして、オブジェクトDの実行が終了した時に、クラス管理オブジェクト１４３及びクラス管理オブジェクト１４５の実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）から、オブジェクトDに割り当てられたスレッド１４４の識別情報が削除される。

【０１７７】このため、名前“ｆｕｎｃｔｉｏｎ－４”の機能が実行されている途中に、名前“ｆｕｎｃｔｉｏｎ－１”の機能の削除要求や追加要求がエージェントクラスオブジェクト１４１に対して行われた場合でも、クラス管理オブジェクト１４３は、自己の実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）を調べることにより、名前“ｆｕｎｃｔｉｏｎ－１”の機能の削除要求や追加要求を保留することが可能となり、名前“ｆｕｎｃｔｉｏｎ－４”の機能が実行されている途中に、名前“ｆｕｎｃｔｉｏｎ－４”の機能の実行が不可能になることを防止することができる。

【０１７８】図１９は、クラスデータ実行処理を示すフローチャートである。ここで、エージェントクラスオブジェクト２２ａ～２２ｎがクラスデータ実行処理を行う場合、ｃａｌｌ（）メソッド２６ｆが呼び出される。

【０１７９】図１９において、まず、エージェントクラスオブジェクト２２ａ～２２ｎに対するアクセス権のチェックが行われ（ステップＳ６１）、アクセス権がなければ処理が中断される。

【０１８０】次に、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”をｃａｌｌ（）メソッド２６ｆの引数から取得し（ステップＳ６２）、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”をクラス管理オブジェクトテーブル２３から検索する（ステップＳ６３）。

【０１８１】次に、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されているどうかを調べ（ステップＳ６４）、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されていない場合、エラーを通知して処理を中断する（ステップＳ６５）。

【０１８２】一方、実行対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されている場合、このクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４に実行中のスレッドＩＤを追加するとともに、このクラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６で示されるクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４に実行中のスレッドＩＤを追加する（ステップＳ６６）。

【０１８３】次に、実行対象のクラスデータからオブジェクトを生成して処理を実行し（ステップＳ６７）、処理が終了するのを待つ（ステップＳ６８）。次に、処理が終わったら、このクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４から実行中のスレッドＩＤを削除するとともに、このクラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６で示されるクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４から実行中のスレッドＩＤを削除する（ステップＳ６９）。

【０１８４】次に、実行中スレッドＩＤリスト４４が空かどうかを調べ（ステップＳ７０）、実行中スレッドＩＤリスト４４が空でない場合、実行処理を終了し（ステップＳ７１）、実行中スレッドＩＤリスト４４が空の場合、削除フラグ４２がオンかどうかを調べる（ステップＳ７２）。削除フラグ４２がオンの場合、削除処理が予約されているため、図２１の削除処理を呼び出した後（ステップＳ７３）、実行処理を終了する（ステップＳ７４）。

【０１８５】一方、削除フラグ４２がオフの場合、置換クラスデータ記憶領域４３が空かどうかを調べ（ステップＳ７５）、置換クラスデータ記憶領域４３が空の場合、実行処理を終了し（ステップＳ７６）、置換クラスデータ記憶領域４３が空でない場合、置換処理が予約されているため、置換クラスデータ記憶領域４３に記憶されているクラスデータを引数にして、図２４及び図２５の置換処理を呼び出した後（ステップＳ７７）、実行処理を終了する。

【０１８６】このように、動作中のエージェントクラスオブジェクト２２ａ～２２ｎがクラスデータの削除要求または置換要求を受け付ける場合に、クラス管理オブジェクト２４ａ～２４ｍが各クラスデータの使用状態を監視することにより、削除対象または置換対象のクラスデータが利用されない状態になるまで、そのクラスデータの削除または置換を延期することが可能となり、動作中のエージェントクラスオブジェクト２２ａ～２２ｎを途中で停止させることなく、クラスデータの削除要求や置換要求を行うことが可能となる。

【０１８７】なお、動作中のエージェントクラスオブジェクト２２ａ～２２ｎにクラスデータの削除要求または置換要求が行われた場合、削除対象または置換対象のクラスデータを使用中のオブジェクトに対して、中止要求の発行し、そのオブジェクトが応答した後にクラスデータの削除を行うようにしてもよい。

【０１８８】次に、エージェントクラスオブジェクト２２ａ～２２ｎからのクラスデータの削除方法について説明する。図２０は、クラスデータ実行中に削除要求があった場合のエージェントクラスオブジェクトの状態を示す図である。

【０１８９】図２０において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト１５１に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル１５５に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト１５３が生成されている。

【０１９０】このクラス管理オブジェクト１５３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１５３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０１９１】また、クラス管理オブジェクト１５３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１５２に登録され、クラス管理オブジェクト１５２の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１５３を特定するポインタがクラス管理オブジェクトテーブル１５２に生成されている。

【０１９２】さらに、名前“ｆｕｎｃｔｉｏｎ－１”の機能が実行されている場合、クラスデータＡ、クラスデータＢ及びクラスデータＣからオブジェクトＡ、オブジェクトＢ及びオブジェクトＣが生成され、スレッド１５４が割り当てられるとともに、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）には、オブジェクトＡ、オブジェクトＢ及びオブジェクトＣに割り当てられたスレッド１５４の識別情報が格納される。

【０１９３】この状態で、名前“ｆｕｎｃｔｉｏｎ－１”が指定され、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する削除要求が行われると、クラス管理オブジェクト１５３は、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）を参照し、クラスデータＡ、クラスデータＢ及びクラスデータＣの使用状況を確認する。

【０１９４】この場合、　実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）には、スレッド１５４の識別情報が格納されているので、クラスデータＡ、クラスデータＢ及びクラスデータＣは使用中であることを認識し、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する削除を保留するするとともに、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）をオンとして、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する削除要求があったことを記憶する。

【０１９５】名前“ｆｕｎｃｔｉｏｎ－１”の機能の実行が終了すると、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）からスレッド１５４の識別情報を削除し、クラス管理オブジェクト１５３は、削除フラグ（ｒｅｍｏｖａｂｌｅｆｌａｇ）の状態を参照することにより、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する削除要求があっかどうかを確認する。

【０１９６】この場合、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）はオンとなっているので、この時点で安全にクラスデータＡ、クラスデータＢ及びクラスデータＣを削除する。

【０１９７】なお、強制的な削除要求であった場合は、実行中の名前“ｆｕｎｃｔｉｏｎ－１”の機能を途中で停止して、クラスデータＡ、クラスデータＢ及びクラスデータＣを直ちに削除するようにしてもよい。

【０１９８】図２１は、クラスデータ削除処理を示すフローチャートである。ここで、エージェントクラスオブジェクト２２ａ～２２ｎがクラスデータ削除処理を行う場合、ｒｅｍｏｖｅ（）メソッド２６ａが呼び出される。

【０１９９】図２１において、まず、エージェントクラスオブジェクト２２ａ～２２ｎに対するアクセス権のチェックが行われ（ステップＳ８１）、アクセス権がなければ処理が中断される。

【０２００】次に、削除対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”を、ｒｅｍｏｖｅ（）メソッド２６ａの引数から取得し（ステップＳ８２）、削除対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”を、クラス管理オブジェクトテーブル２３から検索する（ステップＳ８３）。

【０２０１】次に、削除対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されているどうかを調べ（ステップＳ８４）、削除対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されていない場合、エラーを通知して処理を中断する（ステップＳ８５）。

【０２０２】一方、削除対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されている場合、実行中スレッドＩＤリスト４４が空かどうかを調べ（ステップＳ８６）、実行中スレッドＩＤリスト４４が空でない場合、クラス管理オブジェクト２４ａ～２４ｍの削除フラグ４２をオンにして（ステップＳ８７）、削除処理を終了する（ステップＳ８８）。

【０２０３】一方、実行中スレッドＩＤリスト４４が空の場合、被参照オブジェクトリスト４７が空かどうかを調べ（ステップＳ８９）、被参照オブジェクトリスト４７が空でなければ、エラーを通知して削除処理を中止する（ステップＳ９０）。

【０２０４】一方、被参照オブジェクトリスト４７が空の場合、参照オブジェクトリスト４６が空かどうかを調べ（ステップＳ９１）、参照オブジェクトリスト４６が空でなければ、参照オブジェクトリスト４６で示される各々のクラス管理オブジェクト２４ａ～２４ｍの被参照オブジェクトリスト４７から、このクラス管理オブジェクト２４ａ～２４ｍの被参照情報を削除する（ステップＳ９２）。

【０２０５】一方、参照オブジェクトリスト４６が空の場合、クラス管理オブジェクト２４ａ～２４ｍは、クラス名リスト４５を参照することにより、共有クラスデータテーブル２５から削除するクラスデータを特定し、そのクラス管理オブジェクト２４ａ～２４ｍのクラス名リスト４５に登録されているクラス名のクラスデータを共有クラスデータテーブルから削除する（ステップＳ９３）。

【０２０６】次に、クラス管理オブジェクトテーブル２３から、このクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”を削除して（ステップＳ９４）、削除処理を終了する。

【０２０７】次に、エージェントクラスオブジェクト２２ａ～２２ｎのクラスデータの置換方法について説明する。図２２は、クラスデータ実行中に置換要求があった場合のエージェントクラスオブジェクトの状態を示す図である。ここでは、名前“ｆｕｎｃｔｉｏｎ－１”のクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能を、名前“ｆｕｎｃｔｉｏｎ－２”のクラスデータＤ及びクラスデータＥからなる機能で置き換える例を示している。

【０２０８】図２２において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト１６１に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル１６５に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト１６３が生成されている。

【０２０９】このクラス管理オブジェクト１６３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１６３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフに設定されるとともに、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。また、クラス管理オブジェクト１６３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１６２に登録され、クラス管理オブジェクト１６２の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１６３を特定するポインタがクラス管理オブジェクトテーブル１６２に生成されている。

【０２１０】さらに、名前“ｆｕｎｃｔｉｏｎ－１”の機能が実行されている場合、クラスデータＡ、クラスデータＢ及びクラスデータＣからオブジェクトＡ、オブジェクトＢ及びオブジェクトＣが生成され、スレッド１６４が割り当てられるとともに、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）には、オブジェクトＡ、オブジェクトＢ及びオブジェクトＣに割り当てられたスレッド１６４の識別情報が格納される。

【０２１１】この状態で、名前“ｆｕｎｃｔｉｏｎ－

１”が指定され、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する置換要求が行われると、クラス管理オブジェクト１６３は、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）を参照し、クラスデータＡ、クラスデータＢ及びクラスデータＣの使用状況を確認する。

【０２１２】この場合、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）には、スレッド１６４の識別情報が格納されているので、クラスデータＡ、クラスデータＢ及びクラスデータＣは使用中であることを認識し、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する置換を保留するとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣと置換されるクラスデータＤ及びクラスデータＥを置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）に記憶する。

【０２１３】名前“ｆｕｎｃｔｉｏｎ－１”の機能の実行が終了すると、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）からスレッド１６４の識別情報を削除し、クラス管理オブジェクト１６３は、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）の状態を参照することにより、クラスデータＡ、クラスデータＢ及びクラスデータＣに対する置換要求があっかどうかを確認する。

【０２１４】この場合、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）には、クラスデータＤ及びクラスデータＥが格納されていることから、この時点で安全にクラスデータＡ、クラスデータＢ及びクラスデータＣをクラスデータＤ及びクラスデータＥで置換する。

【０２１５】なお、強制的な削除要求であった場合は、実行中の名前“ｆｕｎｃｔｉｏｎ－１”の機能を途中で停止して、クラスデータＡ、クラスデータＢ及びクラスデータＣを直ちに置換するようにしてもよい。

【０２１６】図２３は、クラスデータの置換後のエージェントクラスオブジェクトの状態を示す図である。図２３において、クラスデータＡ、クラスデータＢ及びクラスデータＣがクラスデータＤ及びクラスデータＥで置き換えられた場合、図２２のクラス管理オブジェクト１６３の置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）からクラスデータＤ及びクラスデータＥが削除されるとともに、図２２のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）に登録されていたクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”が、クラスデータＤ及びクラスデータＥのクラス名“Ｄ”、“Ｅ”で置き換えられ、図２２のクラス管理オブジェクト１６３は、図２３のクラス管理オブジェクト１７３に更新される。

【０２１７】また、図２２の共有クラスデータテーブル１６５に格納されていたクラスデータＡ、クラスデータＢ及びクラスデータＣは、クラスデータＤ及びクラスデータＥで置き換えられ、図２２の共有クラスデータテーブル１６５は、図２３の共有クラスデータテーブル１７４に更新される。

【０２１８】さらに、図２２のクラス管理オブジェクトテーブル１６２に登録されていたクラスデータＡ、クラスデータＢ及びクラスデータＣの名前“ｆｕｎｃｔｉｏｎ－１”は、クラスデータＤ及びクラスデータＥの名前“ｆｕｎｃｔｉｏｎ－２”で置き換えられ、図２２のクラス管理オブジェクトテーブル１６２は、図２３のクラス管理オブジェクトテーブル１７２に更新される。

【０２１９】図２４及び図２５は、クラスデータ置換処理を示すフローチャートである。ここで、エージェントクラスオブジェクト２２ａ～２２ｎがクラスデータ削除処理を行う場合、ｓｅｔ（）メソッド２６ｄが呼び出される。

【０２２０】図２４において、まず、エージェントクラスオブジェクト２２ａ～２２ｎに対するアクセス権のチェックが行われ（ステップＳ１０１）、アクセス権がなければ処理が中断される。

【０２２１】次に、置換対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”及び置換する新たなクラスデータを、ｓｅｔ（）メソッド２６ｄの引数から取得し（ステップＳ１０２）、置換対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”を、クラス管理オブジェクトテーブル２３から検索する（ステップＳ１０３）。

【０２２２】次に、置換対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されているどうかを調べ（ステップＳ１０４）、置換対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されていない場合、エラーを通知して処理を中断する（ステップＳ１０５）。

【０２２３】一方、置換対象のクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍの名前“ｆｕｎｃｔｉｏｎ－１”～“ｆｕｎｃｔｉｏｎ－ｍ”が、クラス管理オブジェクトテーブル２３に登録されている場合、実行中スレッドＩＤリスト４４が空かどうかを調べ（ステップＳ１０６）、実行中スレッドＩＤリスト４４が空でない場合、ｓｅｔ（）メソッド２６ｄの引数から取得したクラスデータを置換クラスデータ記憶領域４３に設定し（ステップＳ１０７）、置換処理を終了する（ステップＳ１０８）。

【０２２４】一方、実行中スレッドＩＤリスト４４が空の場合、被参照オブジェクトリスト４７が空かどうかを調べ（ステップＳ１０９）、被参照オブジェクトリスト

４７が空でなければ、エラーを通知して置換処理を中止する（ステップＳ１１０）。

【０２２５】一方、被参照オブジェクトリスト４７が空の場合、参照オブジェクトリスト４６が空かどうかを調べ（ステップＳ１１１）、参照オブジェクトリスト４６が空でなければ、参照オブジェクトリスト４６で示される各々のクラス管理オブジェクト２４ａ～２４ｍの被参照オブジェクトリスト４７から、このクラス管理オブジェクト２４ａ～２４ｍの被参照情報を削除する（ステップＳ１１２）。

【０２２６】一方、参照オブジェクトリスト４６が空の場合、クラス管理オブジェクト２４ａ～２４ｍは、クラス名リスト４５を参照することにより、共有クラスデータテーブル２５から削除するクラスデータを特定し、そのクラス管理オブジェクト２４ａ～２４ｍのクラス名リスト４５に登録されているクラス名のクラスデータを共有クラスデータテーブルから削除する（ステップＳ１１３）。

【０２２７】次に、図２５に示すように、置換要求のあった新たなクラスデータを共有クラスデータテーブル１２５に書き込む（ステップＳ１１４）。次に、同一のクラス名が既に共有クラスデータテーブル２５に存在しているかどうかを判断し（ステップＳ１１５）、クラスデータの書込みの途中で、同一のクラス名が既に共有クラスデータテーブル２５に存在していた場合、共有クラスデータテーブル２５に新たに格納したクラスデータを削除し、共有クラスデータテーブル２５を元の状態に戻すとともに（ステップＳ１１６）、エラーを通知して処理を終了する（ステップＳ１１７）。

【０２２８】一方、同一のクラス名が既に共有クラスデータテーブル２５に存在していない場合、共有クラスデータテーブル２５に新たに追加した全てのクラスデータのクラス名を、クラス管理オブジェクト２４ａ～２４ｍのクラス名リスト４５に設定する（ステップＳ１１８）。

【０２２９】次に、共有クラスデータテーブル２５に追加したクラスデータに関連するクラスを調べることにより、クラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６に参照先情報を設定するとともに（ステップＳ１１９）、参照先のクラス管理オブジェクト２４ａ～２４ｍの被参照オブジェクトリスト４７に参照元情報を設定して（ステップＳ１２０）、処理を終了する。

【０２３０】次に、エージェントクラスオブジェクト２２ａ～２２ｎでのクラスデータの遠隔実行方法について説明する。図２６は、クラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。

【０２３１】図２６において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト１８１に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル１８５に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト１８３が生成されている。

【０２３２】このクラス管理オブジェクト１８３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト１８３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０２３３】また、クラス管理オブジェクト１８３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル１８２に登録され、クラス管理オブジェクト１８２の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト１８３を特定するポインタがクラス管理オブジェクトテーブル１８２に生成されている。

【０２３４】このエージェントクラスオブジェクト１８１に対し、クラスデータＤ、クラスデータＥ及びクラスデータＡを引数として、ｅｖａｌ（）メソッド１８６が呼ばれた場合、クラス管理オブジェクト１８４が新たに生成される。そして、クラスデータＤ、クラスデータＥ及びクラスデータＡは、このクラス管理オブジェクト１８４の局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）に格納されるとともに、クラス管理オブジェクト１８４のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラスデータＤ、クラスデータＥ及びクラスデータＡのクラス名“Ｄ”、“Ｅ”、“Ａ”がそれぞれ格納される。

【０２３５】クラス管理オブジェクト１８４の削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされる。

【０２３６】このように、エージェントクラスオブジェクト１８１に対し、クラスデータＤ、クラスデータＥ及びクラスデータＡからなる機能を追加する場合、クラスデータＤ、クラスデータＥ及びクラスデータＡを、このクラス管理オブジェクト１８４の局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）に格納することにより、共有クラスデータテーブル１８５にクラスデータ

Aが登録されている場合でも、共有クラスデータテーブル１８５での同一クラス名“A”の衝突を回避することが可能となり、クラスデータD、クラスデータE及びクラスデータAからなる機能を実行させることが可能となる。

【０２３７】このクラスデータD、クラスデータE及びクラスデータAを、クラス管理オブジェクト１８４の局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）に格納する方法は、クラスデータD、クラスデータE及びクラスデータAからなる機能の使用頻度が少ない場合や、実行されるクラスデータが頻繁に変更される場合に有効である。

【０２３８】図２７は、シリアライズデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。図２７において、エージェントクラスオブジェクト１９１の共有クラスデータテーブル１９２にはクラスデータDが格納され、クラスデータDはその属性として、クラスＳｔｒｉｎｇのオブジェクトとクラスＩｎｔｅｇｅｒのオブジェクトとを持つように定義されている。また、エージェントクラスオブジェクト１９１には、クラスデータDのインスタンス１９３が生成されている。

【０２３９】このエージェントクラスオブジェクト１９１が、クラスデータD及びクラスデータDのインスタンス１９３を引数として、エージェントクラスオブジェクト１９４のｅｖａｌ（）メソッド１９５を呼び出す場合、エージェントクラスオブジェクト１９１は、クラスデータDのインスタンス１９３をシリアライズ化する。

【０２４０】そして、クラスデータD及びインスタンス１９３のシリアライズデータをｅｖａｌ（）メソッド１９５の引数に設定し、「“D”｛Ｓｔｒｉｎｇｓ；Ｉｎｔｅｇｅｒｉ；｝、“D”，Ｓｔｒｉｎｇ，“１９：５０AM”，Ｉｎｔｅｇｅｒ，“１９：５０AM”」を送る。

【０２４１】エージェントクラスオブジェクト１９４は、ｅｖａｌ（）メソッド１９５が呼び出されると、クラス管理オブジェクト１９６を生成し、クラスデータDの内容｛Ｓｔｒｉｎｇｓ；Ｉｎｔｅｇｅｒｉ；｝をクラス管理オブジェクト１９６の局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）に格納するとともに、クラス管理オブジェクト１９６のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）にクラスデータDのクラス名“D”を格納する。

【０２４２】また、クラス管理オブジェクト１９６の削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされる。

【０２４３】さらに、クラスデータD及びインスタンス１９３のシリアライズデータに基づいて、エージェントクラスオブジェクト１９１のインスタンス１９３を復元したインスタンス１９７を生成し、クラスデータD及びインスタンス１９７の実行を行う。

【０２４４】このように、エージェントクラスオブジェクト１９４は、クラスデータ及びそのクラスデータから生成されたオブジェクトのシリアライズデータを他のエージェントクラスオブジェクト１９１から受信した場合、受信したクラスデータから新たにオブジェクトを生成するのではなく、受信したクラスデータとシリアライズデータから送信元のオブジェクトの状態を復元して遠隔実行を行うことにより、他のエージェントクラスオブジェクト１９１のインスタンス１９３の実行が可能となる。

【０２４５】図２８は、遠隔実行処理を示すフローチャートである。エージェントクラスオブジェクト２２ａ～２２ｎがクラスデータ遠隔実行処理を行う場合、ｅｖａｌ（）メソッド２６ｅが呼び出される。

【０２４６】図２８において、まず、エージェントクラスオブジェクト２２ａ～２２ｎに対するアクセス権のチェックが行われ（ステップＳ１３０）、アクセス権がなければ処理が中断される。

【０２４７】次に、ｅｖａｌ（）メソッド２６ｅの引数からクラスデータとシリアライズ化されたオブジェクトデータを取得し（ステップＳ１３１）、このクラスデータを管理するクラス管理オブジェクト２４ａ～２４ｍを新たに生成するとともに、このクラス管理オブジェクト２４ａ～２４ｍの局所クラスデータテーブル４１に引数のクラスデータを設定する（ステップＳ１３２）。

【０２４８】次に、局所クラスデータテーブル４１に追加したクラスデータに関連するクラスを調べることにより、そのクラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６に参照先情報を設定するとともに（ステップＳ１３３）、参照先のクラス管理オブジェクト２４ａ～２４ｍの被参照オブジェクトリスト４７に参照元情報を設定する（ステップＳ１３４）。

【０２４９】次に、このクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４に実行中のスレッドＩＤを追加するとともに、このクラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６で示されるクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４に実行中のスレッドＩＤを追加する（ステップＳ１３５）。

【０２５０】次に、ｅｖａｌ（）メソッド２６ｅの引数にシリアライズ化されたオブジェクトデータがあるかどうかを調べ（ステップＳ１３６）、ｅｖａｌ（）メソッド２６ｅの引数にシリアライズ化されたオブジェクトデータがない場合、実行対象のクラスデータからオブジェクトを生成して処理を実行し（ステップＳ１３７）、そ

のオブジェクトの処理が終了するのを待つ（ステップＳ１３９）。

【０２５１】一方、ｅｖａｌ（）メソッド２６ｅの引数にシリアライズ化されたオブジェクトデータがある場合、シリアライズ化されたオブジェクトデータ及び局所クラスデータテーブル４１のクラスデータを用いてそのオブジェクトを復元し、そのオブジェクトの処理を実行する（ステップＳ１３８）。

【０２５２】次に、処理が終わったら、このクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４から実行中のスレッドＩＤを削除するとともに、このクラス管理オブジェクト２４ａ～２４ｍの参照オブジェクトリスト４６で示されるクラス管理オブジェクト２４ａ～２４ｍの実行中スレッドＩＤリスト４４から実行中のスレッドＩＤを削除する（ステップＳ１４０）。

【０２５３】次に、参照オブジェクトリスト４６で示される各々のクラス管理オブジェクト２４ａ～２４ｍの被参照オブジェクトリスト４７から、このクラス管理オブジェクト２４ａ～２４ｍの被参照情報を削除し（ステップＳ１４１）、実行対象のクラスデータを管理していたクラス管理オブジェクト２４ａ～２４ｍを削除する（ステップＳ１４２）。

【０２５４】図２９は、参照関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。図２９において、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト２０１に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル２０６に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト２０３が生成されている。

【０２５５】このクラス管理オブジェクト２０３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト２０３が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）及び参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）が空（ｎｕｌｌ）とされている。

【０２５６】また、クラス管理オブジェクト２０３の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル２０２に登録され、クラス管理オブジェクト２０３の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト２０３を特定するポインタがクラス管理オブジェクトテーブル２０２に生成されている。

【０２５７】さらに、“ｆｕｎｃｔｉｏｎ－４”という名前でクラスデータＤからなる機能がエージェントクラスオブジェクト２０１に追加され、クラスデータＤが共有クラスデータテーブル２０６に格納されるとともに、クラスデータＤを管理するクラス管理オブジェクト２０４が生成されている。

【０２５８】このクラス管理オブジェクト２０４のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト２０４が管理するクラスデータＤのクラス名“Ｄ”が登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｄｅ）が空（ｎｕｌｌ）とされている。

【０２５９】また、クラス管理オブジェクト２０４の名前として、“ｆｕｎｃｔｉｏｎ－４”がクラス管理オブジェクトテーブル２０２に登録され、クラス管理オブジェクト２０４の名前“ｆｕｎｃｔｉｏｎ－４”に対応して、クラス管理オブジェクト２０４を特定するポインタがクラス管理オブジェクトテーブル２０２に生成されている。

【０２６０】ここで、クラスデータＤが、

```
class D0 {
        :
        new A0
        :
        new B0
}
```

のように、例えば、クラスデータＡ及びクラスデータＢの存在を前提としている場合、クラス管理オブジェクト２０４の参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）には、参照先のクラスデータＡ及びクラスデータＢを管理しているクラス管理オブジェクト２０３の参照情報が格納され、クラス管理オブジェクト２０３の被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）には、参照元のクラスデータＤを管理しているクラス管理オブジェクト２０４の被参照情報が格納される。

【０２６１】なお、クラスの呼び出し関係は、例えば、Ｊａｖａ言語の場合、クラスのメソッドであるｇｅｔＤｅｃｌａｒｅｄＣｌａｓｓｅｓ（）を用いることにより調べることができる。

【０２６２】この状態で、エージェントクラスオブジェクト２０１が、名前“ｆｕｎｃｔｉｏｎ－４”を指定してクラスデータＤの遠隔実行の要求を行うと、スレッド２０５が生成され、クラスデータＤを送るとともに、クラス管理オブジェクト２０４の参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）により示されるクラス管理オブジェクト２０３が管理しているクラスデータＡ、クラス

データＢ及びクラスデータＣも一緒に送る。

【０２６３】ここで、クラス管理オブジェクト２０３が管理しているクラスデータＡ、クラスデータＢ及びクラスデータＣと、クラス管理オブジェクト２０４が管理しているクラスデータＤとは、エージェントクラスオブジェクト２０１の共有クラスデータテーブル２０６に格納され、同一のエージェントクラスオブジェクト２０１で管理されていることから、クラスデータＤが参照しているクラスデータＡ及びクラスデータＢを容易に取得することが可能である。

【０２６４】このように、遠隔実行するクラスとそのクラスの実行時に必要となる他のクラスとの呼出し関係をクラスデータの転送前に調べ、遠隔実行するクラスデータを送るときに、そのクラスデータの実行時に必要となる他のクラスデータも一緒に送ることにより、転送先での処理の負担を軽減することができ、特に、分散システムなどのようにクラスデータが分散して格納されている場合に有効である。

【０２６５】図３０は、重複関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。図３０において、例えば、“ｆｕｎｃｔｉｏｎ－５”という名前でクラスデータＡ、クラスデータＤ及びクラスデータＥからなる機能がエージェントクラスオブジェクト２１１に追加され、クラスデータＡ、クラスデータＤ及びクラスデータＥが共有クラスデータテーブル２１４に格納されるとともに、クラスデータＡ、クラスデータＤ及びクラスデータＥを管理するクラス管理オブジェクト２１３が生成されている。

【０２６６】このクラス管理オブジェクト２１３のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト２１３が管理するクラスデータＡ、クラスデータＤ及びクラスデータＥのクラス名“Ａ”、“Ｄ”、“Ｅ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０２６７】また、クラス管理オブジェクト２１３の名前として、“ｆｕｎｃｔｉｏｎ－５”がクラス管理オブジェクトテーブル２１２に登録され、クラス管理オブジェクト２１３の名前“ｆｕｎｃｔｉｏｎ－５”に対応して、クラス管理オブジェクト２１３を特定するポインタがクラス管理オブジェクトテーブル２１２に生成されている。

【０２６８】また、例えば、“ｆｕｎｃｔｉｏｎ－１”という名前でクラスデータＡ、クラスデータＢ及びクラスデータＣからなる機能がエージェントクラスオブジェクト２１５に追加され、クラスデータＡ、クラスデータＢ及びクラスデータＣが共有クラスデータテーブル２１８に格納されるとともに、クラスデータＡ、クラスデータＢ及びクラスデータＣを管理するクラス管理オブジェクト２１７が生成されている。

【０２６９】このクラス管理オブジェクト２１７のクラス名リスト（ｃｌａｓｓ　ｎａｍｅｓ）には、クラス管理オブジェクト２１７が管理するクラスデータＡ、クラスデータＢ及びクラスデータＣのクラス名“Ａ”、“Ｂ”、“Ｃ”がそれぞれ登録され、削除フラグ（ｒｅｍｏｖａｂｌｅ　ｆｌａｇ）がオフとされ、局所クラスデータテーブル（ｌｏｃａｌ　ｃｌａｓｓｅｓ）、置換クラスデータ記憶領域（ｎｅｘｔ　ｃｌａｓｓｅｓ）、実行中スレッドＩＤリスト（ｒｕｎｎｉｎｇ　ｔｈｒｅａｄｓ）、参照オブジェクトリスト（ｒｅｆｅｒｒｉｎｇ）及び被参照オブジェクトリスト（ｒｅｆｅｒｒｅｄ）が空（ｎｕｌｌ）とされている。

【０２７０】また、クラス管理オブジェクト２１７の名前として、“ｆｕｎｃｔｉｏｎ－１”がクラス管理オブジェクトテーブル２１６に登録され、クラス管理オブジェクト２１７の名前“ｆｕｎｃｔｉｏｎ－１”に対応して、クラス管理オブジェクト２１７を特定するポインタがクラス管理オブジェクトテーブル２１６に生成されている。

【０２７１】エージェントクラスオブジェクト２１１が、名前“ｆｕｎｃｔｉｏｎ－５”の機能の遠隔実行をエージェントクラスオブジェクト２１５で行う場合、名前“ｆｕｎｃｔｉｏｎ－５”の機能を構成するクラスデータＤ、クラスデータＥ及びクラスデータＡが、エージェントクラスオブジェクト２１５の共有クラスデータテーブル２１８に存在するかどうかを問い合わせる。

【０２７２】エージェントクラスオブジェクト２１５は、クラスデータＤ、クラスデータＥ及びクラスデータＡが共有クラスデータテーブル２１８に存在するかどうかを調べ、問い合わせを受けたクラスデータＤ、クラスデータＥ及びクラスデータＡのうち、クラスデータＡが共有クラスデータテーブル２１８に存在することを認識する。そして、クラスデータＡの存在をエージェントクラスオブジェクト２１５に応答する。

【０２７３】クラスデータＡの存在を知らされたエージェントクラスオブジェクト２１１は、エージェントクラスオブジェクト２１５に転送するクラスデータＤ、クラスデータＥ及びクラスデータＡからクラスデータＡを除き、クラスデータＤ及びクラスデータＥだけをエージェントクラスオブジェクト２１５に送る。

【０２７４】エージェントクラスオブジェクト２１５は、エージェントクラスオブジェクト２１５から転送されたクラスデータＤ及びクラスデータＥを用いるとともに、自己の共有クラスデータテーブル２１８に格納されているクラスデータＡを用いることにより、名前“ｆｕ

ｎｃｔｉｏｎ－５”の機能を実行する。

【０２７５】このことにより、遠隔実行する際に、転送先の共有クラスデータテーブルに格納されているクラスデータを送る必要がなくなり、クラスデータの転送量を削減することができる。

【０２７６】次に、エージェントクラスオブジェクト２２ａ～２２ｎによるクラスデータの検索方法について説明する。クラス管理オブジェクト２４ａ～２４ｍはクラス検索機能４８を備えており、エージェントクラスオブジェクト２２ａ～２２ｎに追加されたクラスデータからオブジェクトを生成するときや、オブジェクトのメソッド内部で宣言されているクラスデータからオブジェクトを生成する時には、クラス管理オブジェクト２４ａ～２４ｍのｌｏａｄＣｌａｓｓ（）メソッドを呼び出す。

【０２７７】図３１は、クラス検索処理を示すフローチャートである。図３１において、まず、ｌｏａｄＣｌａｓｓ（）メソッドの引数から検索対象のクラスデータのクラス名を取得し（ステップＳ１５１）、局所クラスデータテーブル４１を検索する（ステップＳ１５２）。そして、局所クラスデータテーブル４１の検索結果を調べ（ステップＳ１５３）、検索対象のクラスデータが見つかった場合、そのクラスデータを戻り値として返す（ステップＳ１５８）。

【０２７８】一方、検索対象のクラスデータが局所クラスデータテーブル４１に見つからなかった場合、共有クラスデータテーブル２５を検索する（ステップＳ１５４）。そして、共有クラスデータテーブル２５の検索結果を調べ（ステップＳ１５５）、検索対象のクラスデータが見つかった場合、そのクラスデータを戻り値として返す（ステップＳ１５８）。

【０２７９】一方、検索対象のクラスデータが共有クラスデータテーブル２５に見つからなかった場合、エージェントクラスライブラリ３１及びシステムクラスライブラリ３２を検索する（ステップＳ１５６）。そして、エージェントクラスライブラリ３１及びシステムクラスライブラリ３２の検索結果を調べ（ステップＳ１５７）、検索対象のクラスデータが見つかった場合、そのクラスデータを戻り値として返す（ステップＳ１５８）。

【０２８０】一方、検索対象のクラスデータがエージェントクラスライブラリ３１及びシステムクラスライブラリ３２に見つからなかった場合、エラーを通知して終了する（ステップＳ１５９）。

【０２８１】以上説明したように、本発明の実施例によれば、分散オブジェクトにクラスデータの置換機能と遠隔実行機能を付加したエージェントクラスを用意することで、柔軟な分散オブジェクトシステムの運用が可能となり、開発者にクラス管理の知識がなくても容易に安全にこれらの機能を利用することが可能となる。

【０２８２】以上、本発明の実施例について説明したが、本発明は上述した実施例に限定されることなく、本発明の技術的思想の範囲内で他の様々の変更が可能である。

【０２８３】

【発明の効果】以上説明したように、本発明によれば、プログラムの使用状況を追加されたクラスデータごとに監視することにより、プログラムが動作している場合にどのクラスデータが使用されているかを判別することが可能となることから、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【０２８４】また、本発明の一態様によれば、動作中のプログラムに対して削除や置換などの処理要求が行われた場合においても、そのプログラムの動作を停止させることなく、そのプログラムに対する削除や置換などの処理を行わせることが可能となり、プログラムのカスタマイズ時におけるプログラマの負担を軽減することが可能となる。

【０２８５】また、本発明の一態様によれば、クラスデータの使用状況を監視する監視オブジェクトを監視対象のオブジェクト内に設けることにより、監視対象となるプログラム自体にそのプログラムを監視する機能を容易に付加することが可能となり、プログラマにクラス管理の知識がなくても、プログラムの使用状況の監視を容易に行うことが可能となるとともに、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【０２８６】また、本発明の一態様によれば、クラスデータの追加要求が行われるごとに監視オブジェクトを生成することにより、プログラムに新たに追加された機能ごとにプログラムの監視を行うことが可能となり、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となることから、プログラムの一部の機能を特定するだけで、そのプログラムの機能の削除や置換をそのプログラムの動作中に行うことが可能となる。

【０２８７】また、本発明の一態様によれば、クラス管理オブジェクトの識別情報を記憶することにより、クラス管理オブジェクトに与えられた名前などの識別情報を指定するだけで、プログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となり、プログラムをカスタマイズする際のプログラマの負担を軽減することが可能となる。

【０２８８】また、本発明の一態様によれば、クラス管理オブジェクトに固有のクラスデータを格納する局所クラスデータテーブルを設けることにより、他のオブジェクトから送られてきたクラスデータとクラス管理オブジェクトが既に管理しているクラスデータとの間に衝突が発生した場合においても、他のオブジェクトから送られてきたクラスデータをクラス管理オブジェクトが既に管理しているクラスデータと独立に保有することが可能と

なり、エージェントクラスオブジェクトは、他のオブジェクトから送られてきたクラスデータを遠隔実行することが可能となる。

【０２８９】また、本発明の一態様によれば、クラスデータに対する削除要求を記憶しておくことにより、削除要求されたクラスデータが使用中の場合においても、そのクラスデータに対する削除要求を受け入れることが可能となることから、クラスデータに対する削除要求が拒否されたために、そのクラスデータに対する削除要求を改めて行う必要がなくなり、プログラマがクラスデータを削除する際の負担を軽減することが可能となる。

【０２９０】また、本発明の一態様によれば、置換要求時の新たなクラスデータを記憶しておくことにより、置換要求されたクラスデータが使用中の場合においても、そのクラスデータに対する置換要求を受け入れることが可能となることから、クラスデータに対する置換要求が拒否されたために、そのクラスデータに対する置換要求を改めて行う必要がなくなり、プログラマがクラスデータを置換する際の負担を軽減することが可能となる。

【０２９１】また、本発明の一態様によれば、クラスデータを実行しているスレッドを記憶することにより、実行中スレッドリストを参照するだけで、実行されているクラスデータが存在するかどうかを判別することができ、クラスデータの使用状況を容易に把握することが可能となる。

【０２９２】また、本発明の一態様によれば、追加されたクラスデータに対応するクラス名を記憶することにより、クラス管理オブジェクトが管理しているクラスデータを容易に把握することが可能となり、追加されたクラスデータを単位としてクラスデータに対する処理を行うことが可能となる。

【０２９３】また、本発明の一態様によれば、クラスデータの参照関係をクラス管理オブジェクトに記憶することにより、クラスデータの参照関係を考慮しながらクラスデータに対する処理を行うことが可能となり、プログラマがクラスデータの参照関係を認識する必要がなくなることから、プログラムのカスタマイズを容易に行うことが可能となる。

【０２９４】また、本発明の一態様によれば、他のエージェントクラスオブジェクトからのアクセス権をチェックするアクセス制御クラスを設けることにより、エージェントクラスオブジェクトは、他のエージェントクラスオブジェクトからのアクセスがあった場合、アクセス制御クラスに基づいてアクセス制御オブジェクトを生成することが可能となることから、エージェントクラスオブジェクトからのアクセスがある度に、アクセス権のチェックを行うことが可能となるとともに、エージェントクラスオブジェクトごとにアクセス条件を容易に変更することが可能となり、プログラムコードの追加・更新のインターフェイスのセキュリティを向上させることが可能となる。

【０２９５】また、本発明の一態様によれば、クラスデータのクラス名の重複を排除することにより、共有クラスデータテーブル内で同一のクラスの衝突が発生することを防止することができる。

【０２９６】また、本発明の一態様によれば、参照元のクラスデータの実行中に参照先のクラスデータの削除要求や置換要求がなされた場合、参照元のクラスデータの実行が終了するまで参照先のクラスデータの削除や置換を一時的に保留しておくことにより、クラスデータの実行が途中で不可能になることを防止することが可能となる。

【０２９７】また、本発明の一態様によれば、削除要求や置換要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を削除したり、置換したりすることにより、プログラムの一部の機能を削除または置換する際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【０２９８】また、本発明の一態様によれば、他のエージェントクラスオブジェクトから送られたクラスデータを局所的に格納することにより、クラス管理オブジェクトに対応して既に格納されているクラスデータとの衝突を回避することが可能となる、エージェントクラスオブジェクトによる遠隔実行を円滑に行うことが可能となる。

【０２９９】また、本発明の一態様によれば、クラスデータ及びクラスデータから生成されたオブジェクトのシリアライズデータに基づいて、そのクラスデータから生成されたオブジェクトを復元するようにしている。

【０３００】このことにより、クラスデータから生成されたオブジェクトをシリアライズ化して送信し、シリアライズデータを受信先で復元することにより、オブジェクトの遠隔実行を行うことが可能となる。

【０３０１】また、本発明の一態様によれば、実行対象のクラスデータが参照しているクラスデータを送信することにより、参照関係を有しているクラスデータについても、遠隔実行を行うことが可能となる。

【０３０２】また、本発明の一態様によれば、送信先のエージェントクラスオブジェクトに存在しないクラスデータだけを送信することにより、遠隔実行を行う際の通信量を減らすことが可能となり、遠隔実行の速度を上げることが可能となる。

【図面の簡単な説明】

【図１】本発明の一実施例に係わる情報処理装置の機能的な構成を示すブロック図である。

【図２】図１のクラスデータ処理手段の構成例を示すブロック図である。

【図３】本発明の一実施例に係わるコンピュータの機能

的な構成を示すブロック図である。
【図４】図３のクラス管理オブジェクトの構成例を示す図である。
【図５】エージェントクラスの構成例を示す図である。
【図６】クラスデータ追加時のクラス管理オブジェクトの生成方法を示す図である。
【図７】テンプレートのコード化の例を示す図である。
【図８】図３のコンピュータのシステム構成例を示す図である。
【図９】図３の通信デーモンの動作を示すフローチャートである。
【図１０】図３の通信スレッドの動作を示すフローチャートである。
【図１１】クラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。
【図１２】参照関係のあるクラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。
【図１３】重複するクラスデータをエージェントクラスオブジェクトから削除する方法を示す図である。
【図１４】クラスデータ追加処理を示すフローチャートである。
【図１５】アクセス権チェック処理を示すフローチャートである。
【図１６】クラスデータ取得処理を示すフローチャートである。
【図１７】クラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。
【図１８】参照関係があるクラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。
【図１９】クラスデータ実行処理を示すフローチャートである。
【図２０】クラスデータ実行中に削除要求があった場合のエージェントクラスオブジェクトの状態を示す図である。
【図２１】クラスデータ削除処理を示すフローチャートである。
【図２２】クラスデータ実行中に置換要求があった場合のエージェントクラスオブジェクトの状態を示す図である。
【図２３】クラスデータの置換後のエージェントクラスオブジェクトの状態を示す図である。
【図２４】クラスデータ置換処理を示すフローチャートである。
【図２５】クラスデータ置換処理を示すフローチャート（続き）である。
【図２６】クラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。
【図２７】シリアライズデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。
【図２８】遠隔実行処理を示すフローチャートである。
【図２９】参照関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。
【図３０】重複関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。
【図３１】クラス検索処理を示すフローチャートである。

【符号の説明】
１　クラスデータ記憶手段
２　クラスデータ監視手段
３　クラスデータ処理手段
１１　受付手段
１２　判定手段
１３　延期手段
２１　コンピュータ
２２ａ～２２ｎ、１０１、１１１、１２１、１３１、１４１、１５１、１６１、１７１、１８１、１９１、１９４、２０１、２１１、２１５、３０１　エージェントクラスオブジェクト
２３、５３、１０２、１１２、１２２、１３２、１４２、１５２、１６２、１７２、１８２、２０２、２１２、２１６　クラス管理オブジェクトテーブル
２４ａ～２４ｍ、５４、１０３、１１３、１１４、１２３、１３３、１４３、１４５、１５３、１６３、１７３、１８３、１８４、１９６、２０３、２０４、２１３、２１７、３０３、３０４、３０５　クラス管理オブジェクト
２５、５５、１０４、１１５、１２４、１３５、１４７、１５５、１６５、１７４、１８５、１９２、２０６、２１４、２１８、３０２　共有クラスデータテーブル
２６ａ～２６ｆ、５６、１２５、１８６、１９５　外部公開メソッド
２７ａ～２７ｋ、１３４、１４４、１４６、１５４、１６４、２０５　通信スレッド
２８　通信デーモン
２９　ネットワークソケットＡＰＩ
３０　オペレーティングシステム
３１　エージェントクラスライブラリ
３２　システムクラスライブラリ
４１　局所クラスデータテーブル
４２　削除フラグ
４３　置換クラスデータ格納部
４４　実行中スレッドＩＤリスト
４５　クラス名リスト
４６　参照オブジェクトリスト
４７　被参照オブジェクトリスト
５１　クラスライブラリ
５２　基本クラス

55

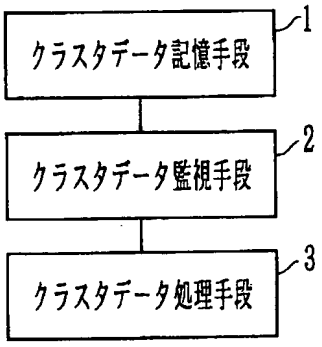
５７　サブクラス
５８ａ～５８ｊ、１９３、１９７　インスタンス
６１　ＣＰＵ
６２　ＲＡＭ
６３　ＲＯＭ
６４　通信インターフェイス
６５　プリンタ
６６　ディスプレイ
６７　キーボード
６８　マウス

56

６９　ドライバ
７０　ハードディスク
７１　フロッピーディスク
７２　磁気テープ
７３　光ディスク
７４　ＩＣメモリカード
８１　初期メニュー
８２　ホップアップメニュー
８３　フィールド欄
８４　内容欄

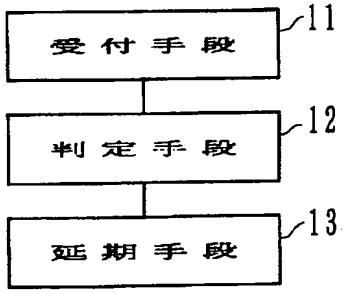
【図１】

本発明の一実施例に係わる
情報処理装置の機能的な構成を示すブロック図



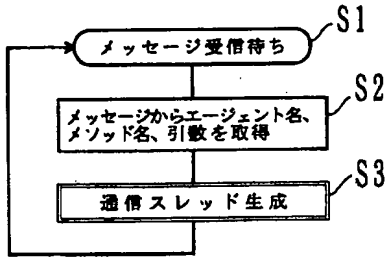
【図２】

※１のクラスデータ処理手段の
構成例を示すブロック図



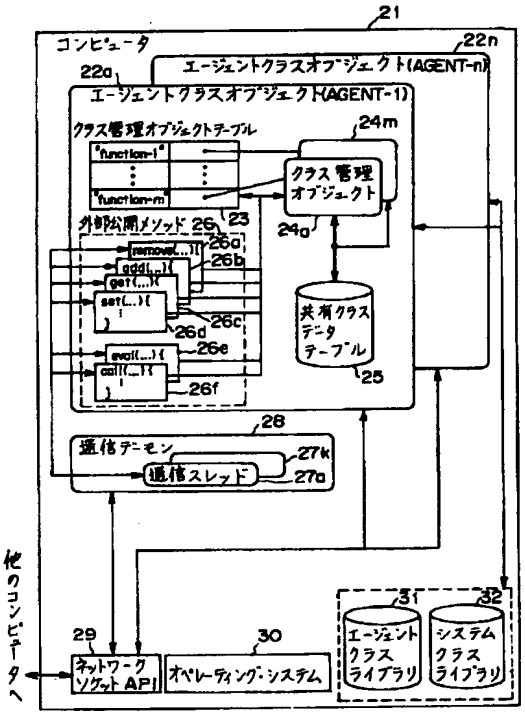
【図９】

■３の通信デーモンの動作を示すフローチャート



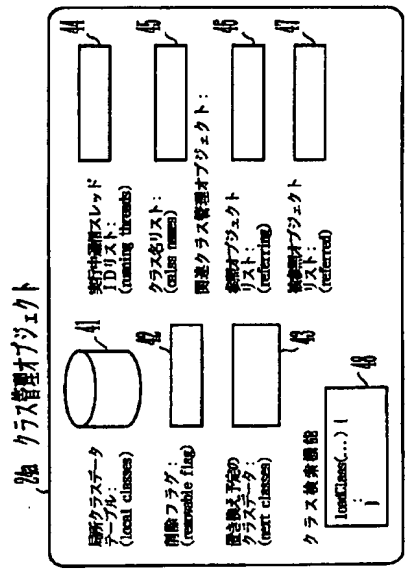
【図３】

本発明の一実施例に係わるコンピュータの機能的な構成を示すブロック図



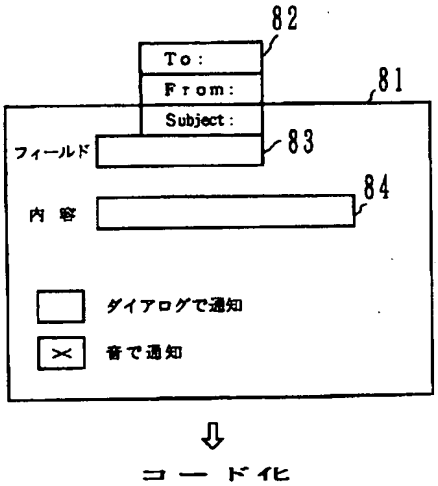
【図４】

図３のクラス管理オブジェクトの構成例を示す図



【図７】

テンプレートのコード化の例を示す図

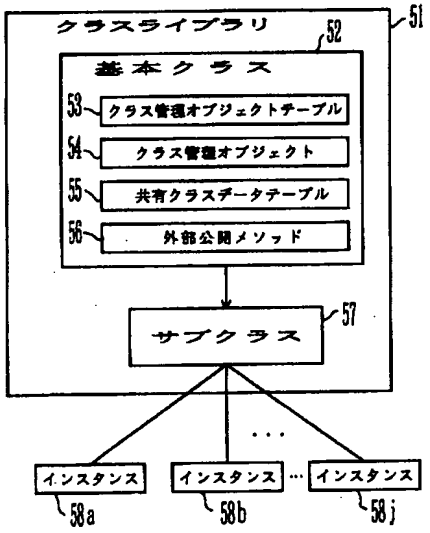


⇩

コード化

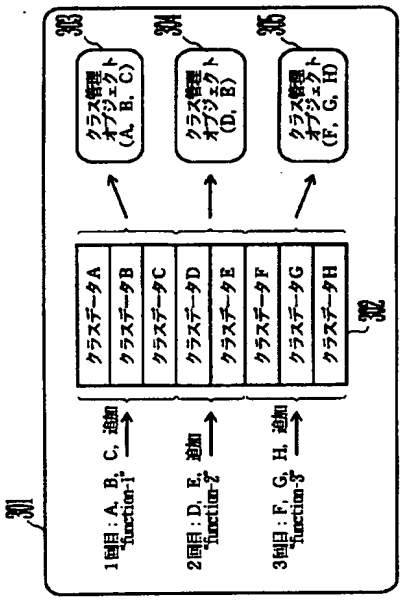
【図５】

エージェントクラスの構成例を示す図



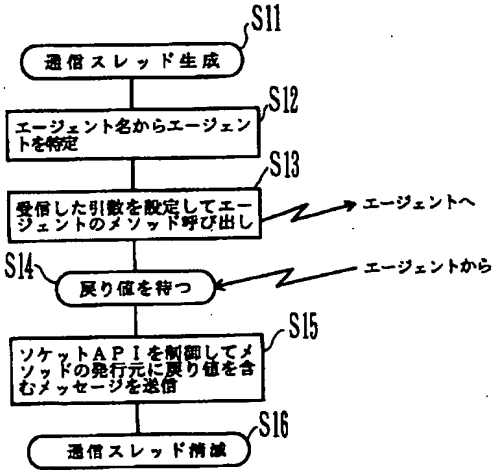
【図６】

クラスデータ追加時の
クラス管理オブジェクトの生成方法を示す図



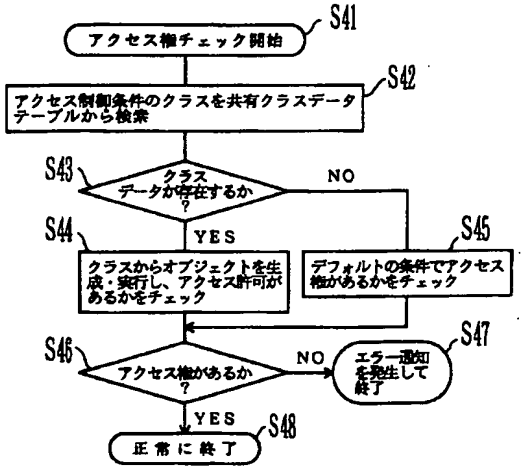
【図１０】

図３の通信スレッドの動作を示すフローチャート



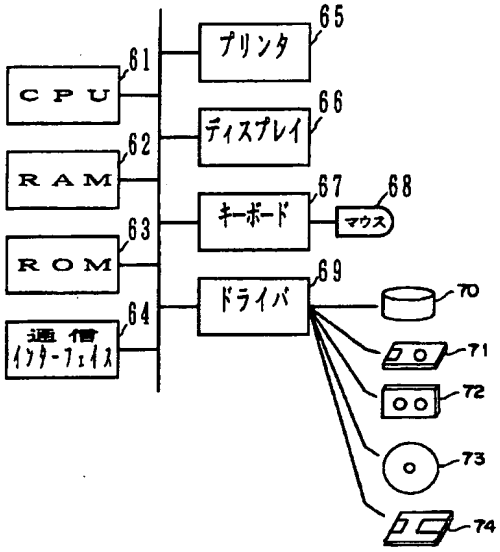
【図１５】

アクセス権チェック処理を示すフローチャート



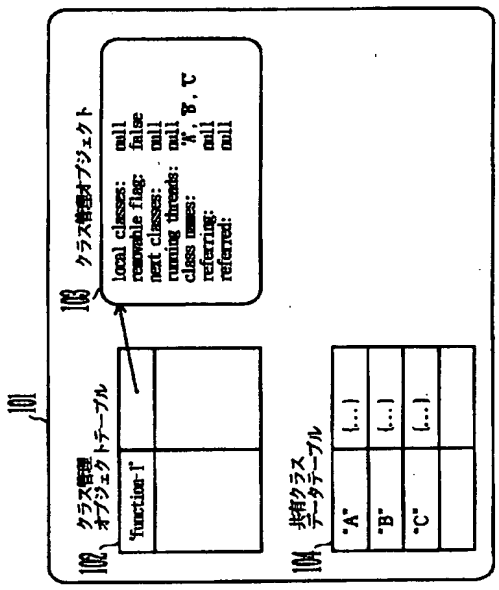
【図８】

図３のコンピュータのシステム構成例を示す図



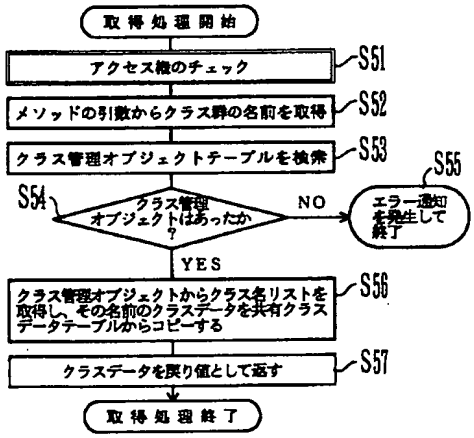
【図１１】

クラスデータをエージェントクラスオブジェクトに追加した状態を示す図



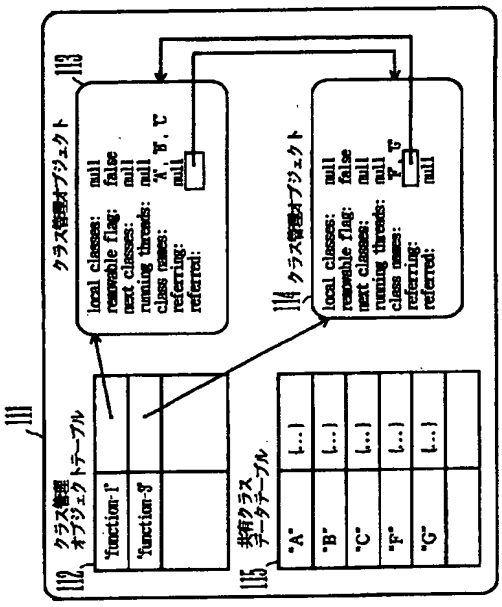
【図１６】

クラスデータ取得処理を示すフローチャート



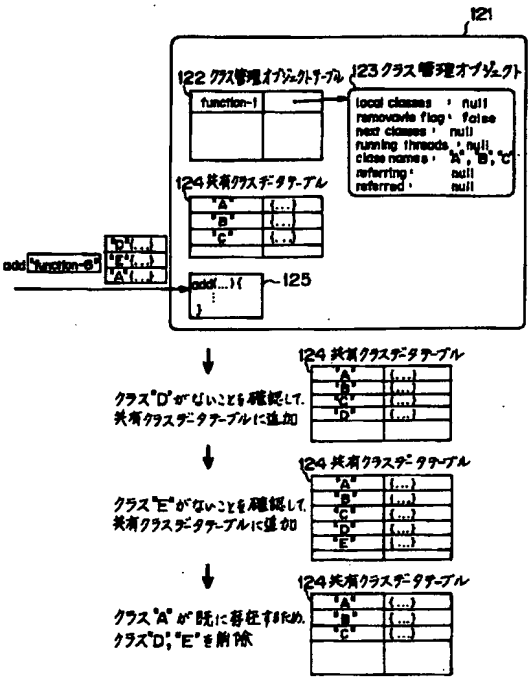
【図１２】

参照関係のあるクラスデータをエージェントクラスオブジェクトに追加した状態を示す図



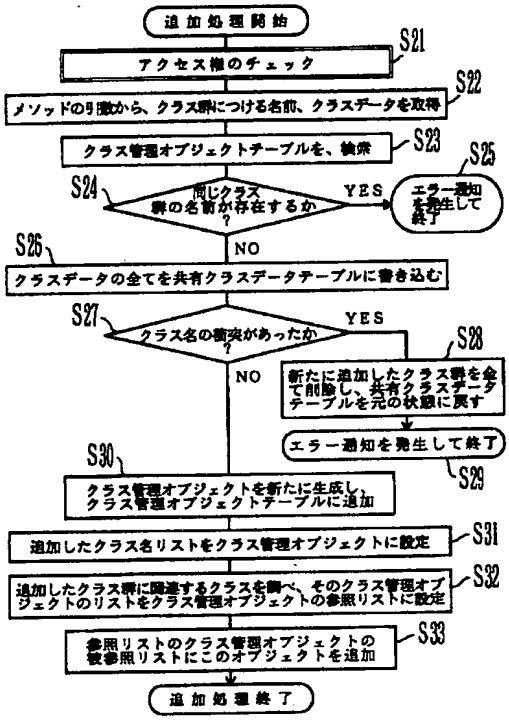
【図１３】

重複するクラスデータをエージェントクラスオブジェクトから削除する方法を示す図



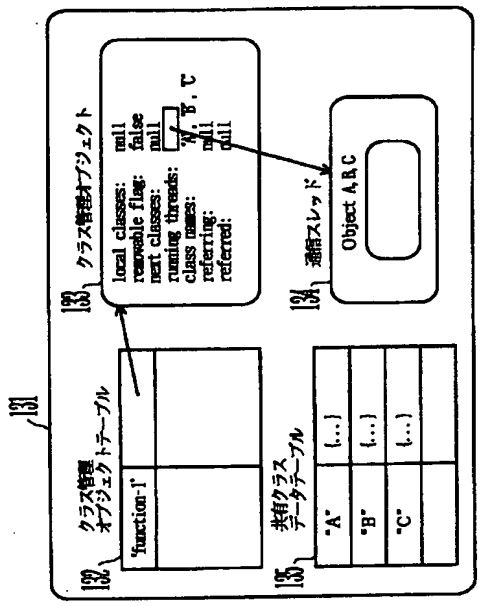
【図１４】

クラスデータ追加処理を示すフローチャート



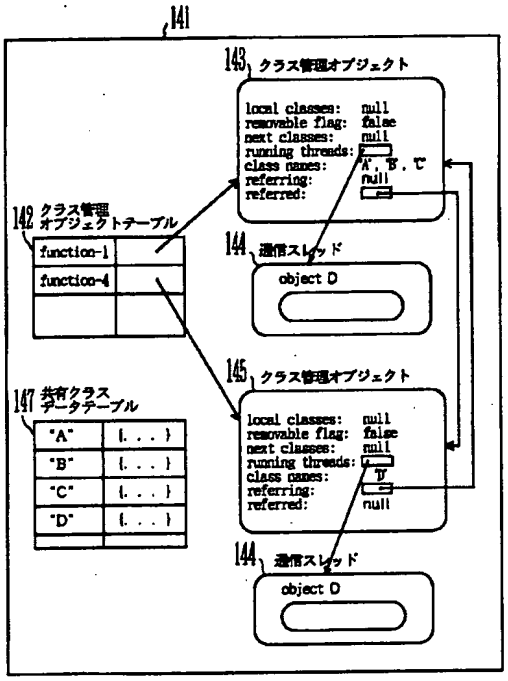
【図１７】

クラスデータ実行中のエージェントクラスオブジェクトの状態を示す図



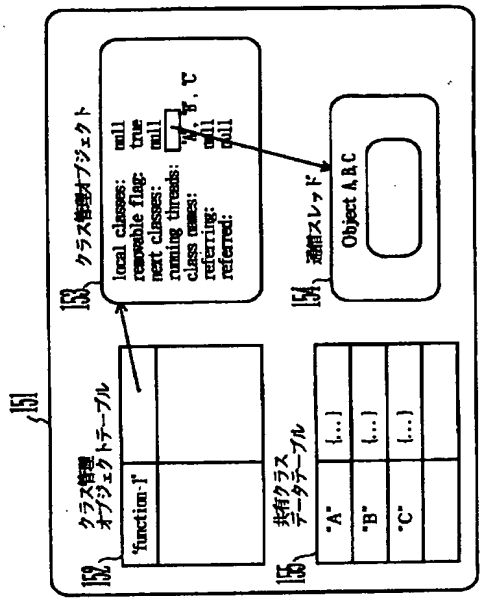
【図１８】

参照関係があるクラスデータ実行中の
エージェントクラスオブジェクトの状態を示す図



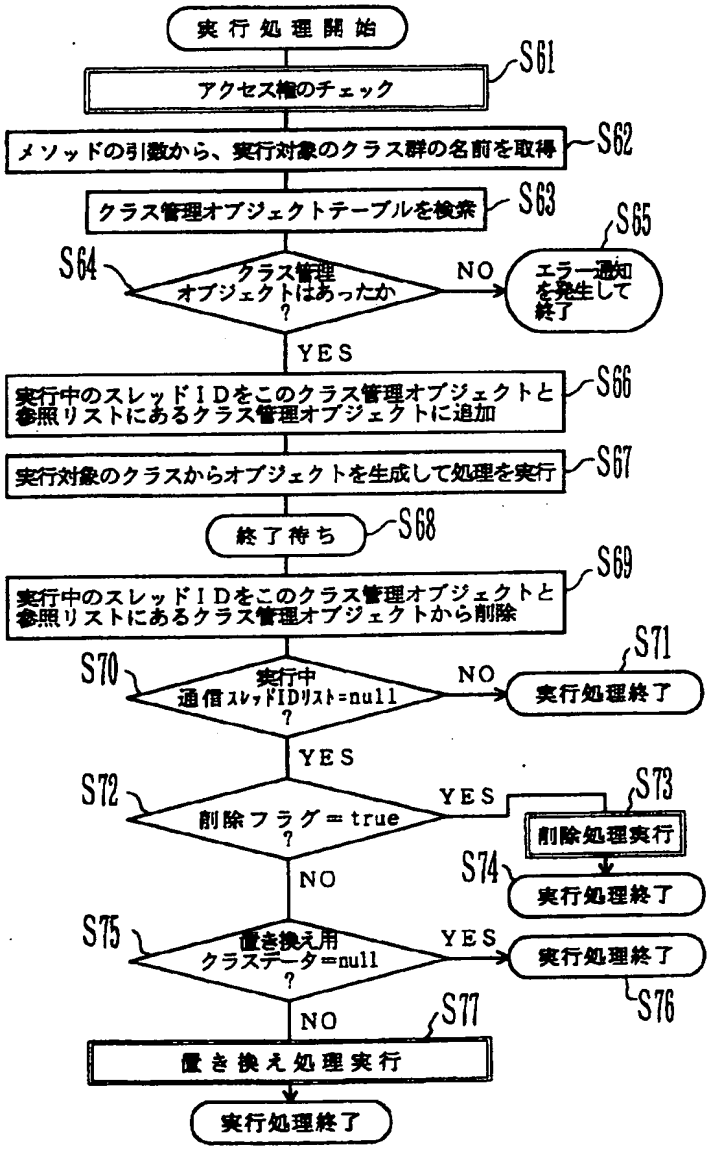
【図２０】

クラスデータ実行中に削除要求があった場合の
エージェントクラスオブジェクトの状態を示す図



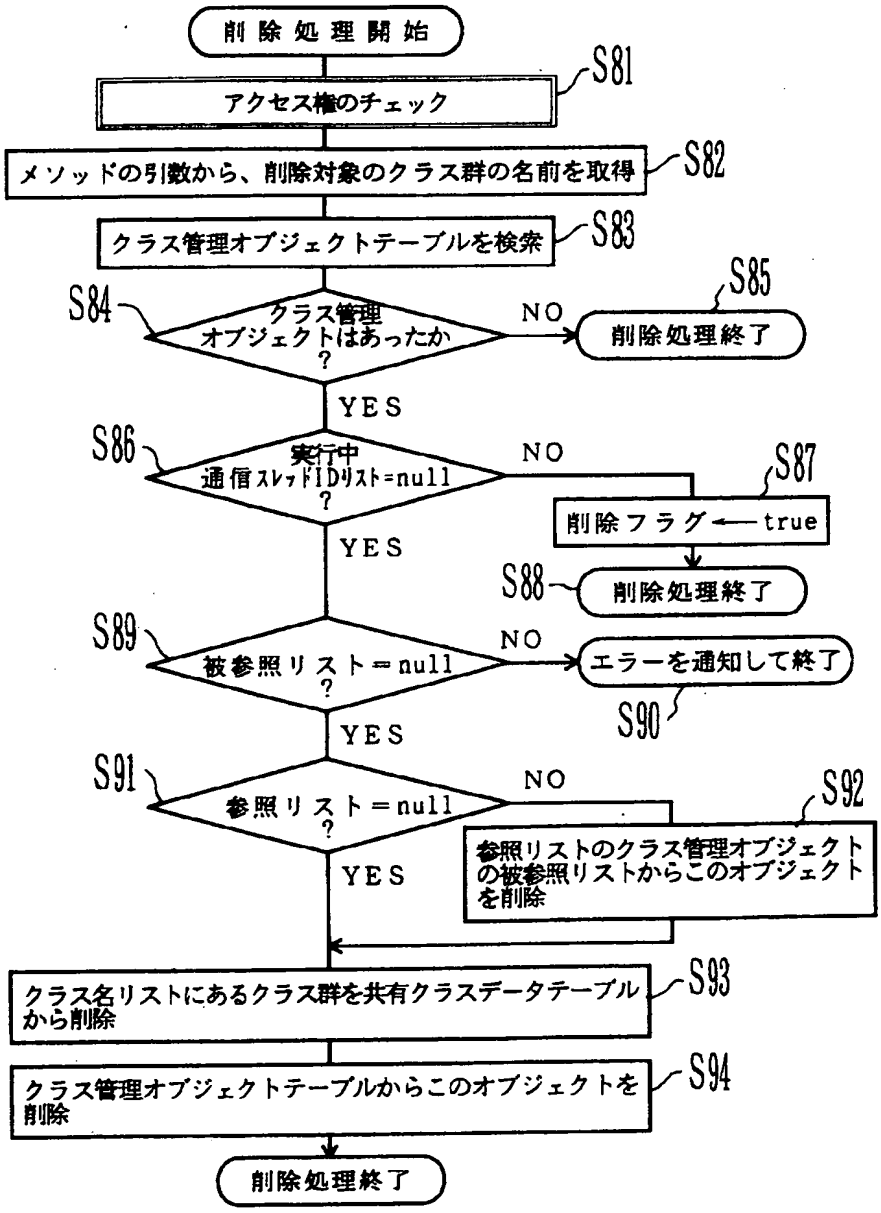
【図１９】

クラスデータ実行処理を示すフローチャート



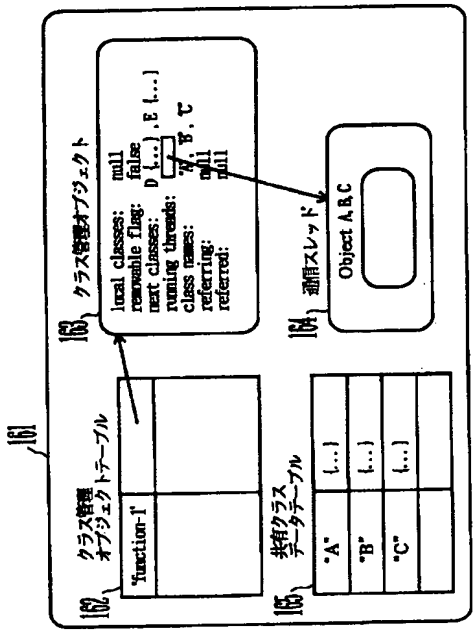
【図２１】

クラスデータ削除処理を示すフローチャート



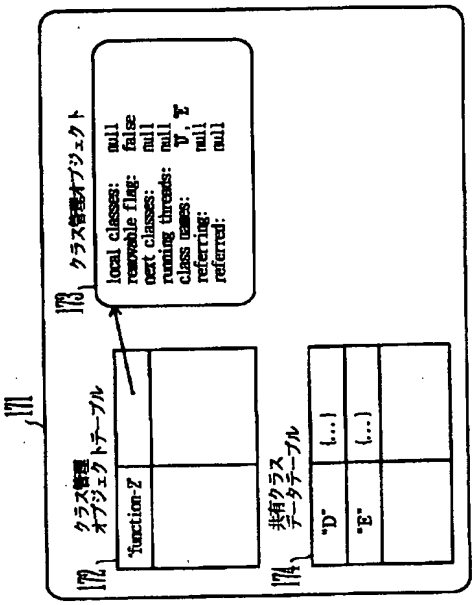
【図２２】

クラスデータ実行中に置換要求があった場合のエージェントクラスオブジェクトの状態を示す図



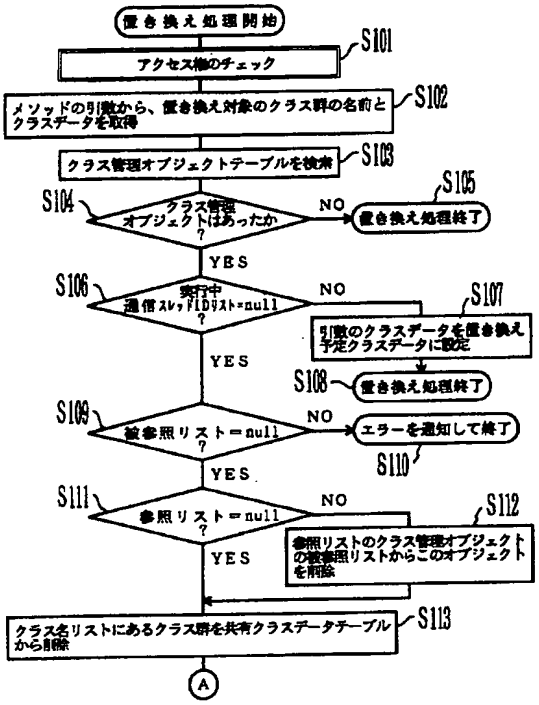
【図２３】

クラスデータ置換後のエージェントクラスオブジェクトの状態を示す図



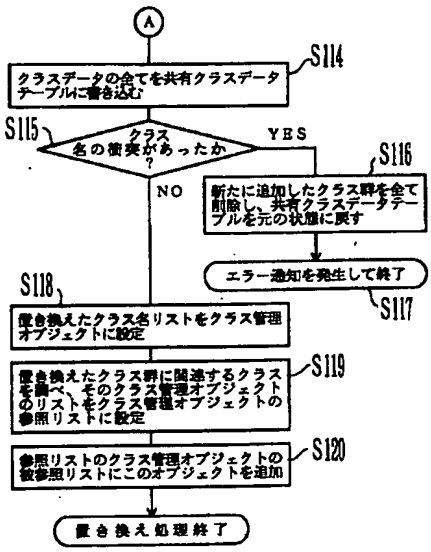
【図２４】

クラスデータ置換処理を示すフローチャート



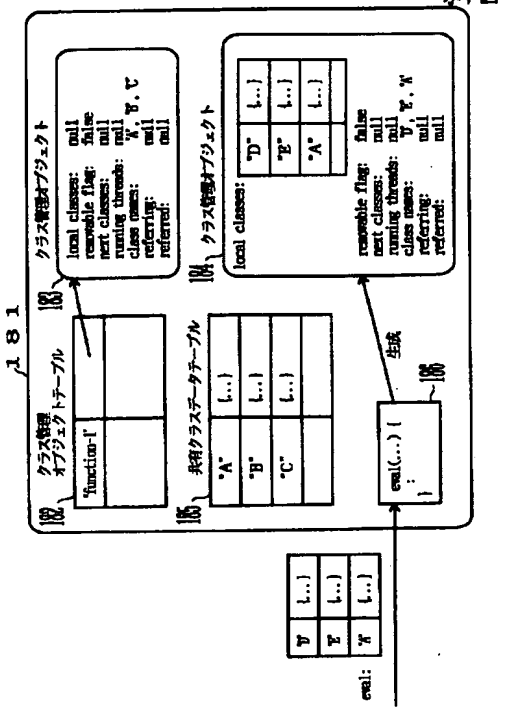
【図２５】

クラスデータ置換処理を示すフローチャート（続き）



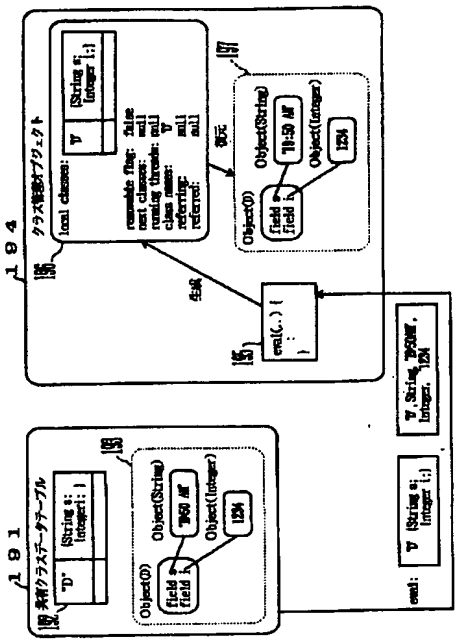
【図２６】

クラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図



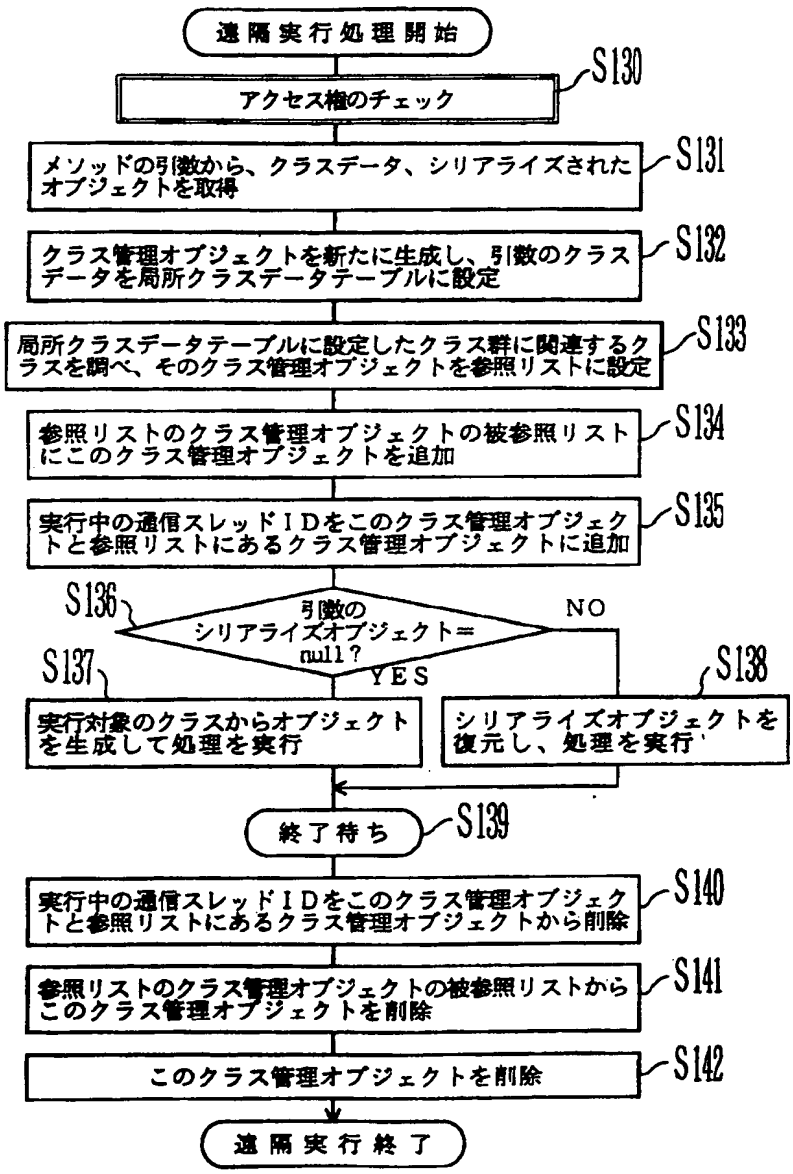
【図２７】

シリアライズデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図



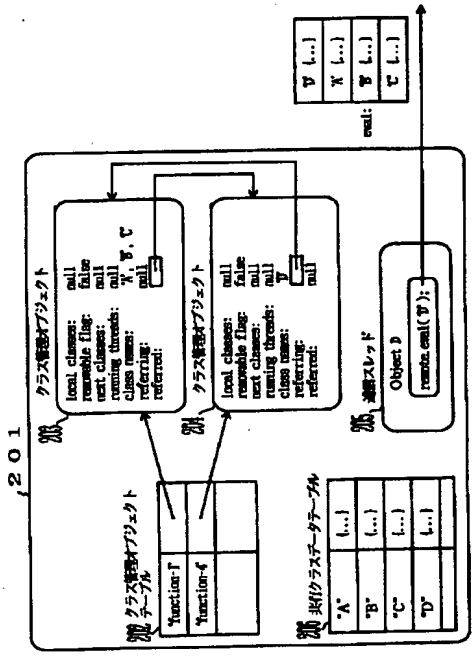
【図２８】

遠隔実行処理を示すフローチャート



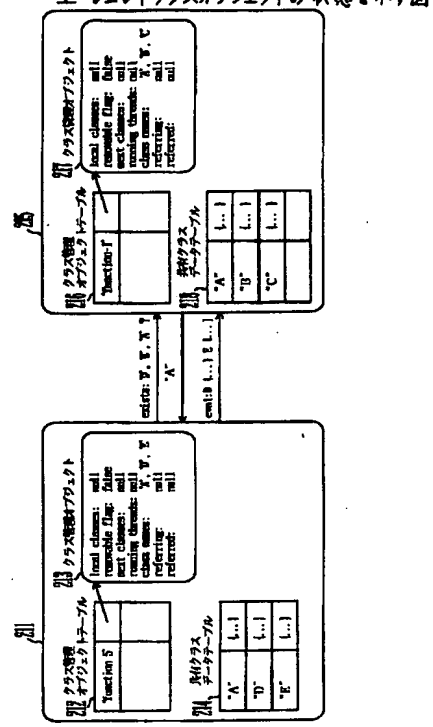
【図２９】

参照関係を有するクラスデータの遠隔実行時の
エージェントクラスオブジェクトの状態を示す図



【図３０】

重複関係を有するクラスデータの遠隔実行時の
エージェントクラスオブジェクトの状態を示す図



【図３１】

クラス検索処理を示すフローチャート